CARRERA:

INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

PREVIO AL GRADO ACADÉMICO DE INGENIERÍA
DE SISTEMAS Y COMPUTACIÓN

TEMA DE INVESTIGACIÓN:

ANÁLISIS DE TÉCNICAS DE INYECCIÓN DE FALLOS EN SISTEMAS DISTRIBUIDOS

LÍNEA DE INVESTIGACIÓN:

PROGRAMACIÓN Y DESARROLLO DE SOFTWARE

PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERÍA DE SISTEMA Y COMPUTACIÓN

AUTOR:

LUIS ERNESTO TORRES FARÍAS

ASESOR:

JOSÉ LUIS CARVAJAL

Esmeraldas, 2021

TRIBUNAL DE GRADUACIÓN

Título: Análisis de técnicas de inyección de fallos en sistemas distribuidos.

Autor(a): Luis Torres Farías

Mgt. José Luis Carvajal **f.**_____
Asesor

Mgt. Susana Patiño **f.**_____
Lector #1

Mgt. Xavier Quiñonez **f.**_____
Lector #2

Mgt. Susana Patiño **f.**_____
Coordinadora de carrera

AUTORÍA

Yo, **Luis Ernesto Torres Farías** con número de cédula de identidad 0803594951 manifiesto que mediante la presente investigación sobre el tema **"ANÁLISIS DE TÉCNICAS DE INYECCIÓN DE FALLOS EN SISTEMAS DISTRIBUIDOS"** los resultados obtenidos como tesis de grado, previo a la obtención del título de **"INGENIERO EN SISTEMAS Y COMPUTACIÓN"** son de total responsabilidad del autor, y que se ha respetado las fuentes de información consultadas, realizando las citas correspondientes y los resultados alcanzados son totalmente personales, únicos y legítimos. Al mismo tiempo declaro que todo el contenido incluyendo resultados, discusión, conclusiones, recomendaciones y otros efectos legales y académicos que se desglosan, son y serán exclusiva responsabilidad legal y académica del autor y de la PUCESE.

Índice de contenidos

Índice de figuras

Índice de tablas

RESUMEN

Esta investigación abordó el tema de inyecciones de fallos, debido a la necesidad de crear sistemas distribuidos con un alto grado de fiabilidad, los cuales por la importancia del servicio que prestan, un detenimiento de estos puede resultar en pérdidas significativas, ya sea en lo económico, a nivel de servicios o en el peor de los casos pérdidas humanas.

El objetivo principal de la investigación fue realizar un estudio documental de las principales técnicas de inyección de fallos aplicables en sistemas distribuidos para establecer buenas prácticas a desarrolladores permitiéndoles crear sistemas con un mayor grado de fiabilidad. Para conseguir este objetivo se planteó una metodología mixta fundamentada en los enfoques cualitativo y cuantitativo, así mismo, se aplicó un mapeo sistemático para la recolección y categorización de datos en conjunto con la técnica de estadística descriptiva para resumir la información.

De acuerdo con el análisis realizado, se concluyó que existen cinco técnicas que se pueden aplicar en el desarrollo de un sistema, también, se describe un conjunto de herramientas identificadas en la bibliografía las cuales hacen posible estas implementaciones, por otra parte, se compararon las técnicas en base a parámetros de tipo cualitativos y cuantitativos (genericidad, accesibilidad y controlabilidad, neutralidad, automatización, precisión y coste).

Los resultados permitieron apreciar las cualidades de mayor importancia de cada técnica aplicada, en base a ponderaciones (alta, media y baja) por cada parámetro antes mencionado. Finalmente, se presentó un esquema de la taxonomía propuesta con lo cual se alcanza el objetivo antes planteado.

Palabras claves: fiabilidad, sistema distribuido, genericidad, controlabilidad, taxonomía, inyección de fallos.

ABSTRACT

This research addressed the issue of fault injections, due to the need to create distributed systems with a high degree of reliability, which due to the importance of the service they provide, a stoppage of these can result in significant losses, either economically, at the level of services or in the worst-case human losses.

The main objective of the research was to carry out a documentary study of the main fault injection techniques applicable in distributed systems to establish good practices for developers, allowing them to create systems with a higher degree of reliability. To achieve this objective, a mixed methodology based on qualitative and quantitative approaches was used, as well as a systematic mapping for the collection and categorization of data together with the descriptive statistics technique to summarize the information.

According to the analysis carried out, it was concluded that there are five techniques that can be applied in the development of a system, also, a set of tools identified in the bibliography is described which make these implementations possible, on the other hand, the techniques were compared based on qualitative and quantitative parameters (genericity, accessibility and controllability, neutrality, automation, precision and cost).

The results showed the most important qualities of each technique applied, based on weightings (high, medium and low) for each of the aforementioned parameters. Finally, an outline of the proposed taxonomy was presented, thus achieving the aforementioned objective.

Keywords: reliability, distributed system, genericity, controllability, taxonomy, fault injection.

INTRODUCCIÓN

PRESENTACIÓN DEL PROBLEMA

Los sistemas distribuidos representan un papel cada vez más notorio en la cotidianidad de las personas debido al crecimiento tecnológico de forma exponencial que se viene experimentando. Es común que los servicios y comodidades que se utilizan a diario estén controlados por sistemas distribuidos dado el caso de un sistema bancario, sistemas de transportes, sistemas industriales, sistemas militares, por mencionar varios casos donde la disponibilidad es un factor de suma importancia y una falla puede significar la suspensión de los servicios primordiales, pérdidas económicas o en el peor de los casos pérdidas humanas. Por tal motivo en el proceso de desarrollo de un sistema, para que este sea tolerante a fallos se requiere que sea validado, o medido según sus parámetros de confiabilidad.

En este trabajo se presenta la inyección de fallos como una técnica sustancial en la verificación de sistemas informáticos. Para ello se detallan conceptos básicos relacionados a la categorización de fallos, varias técnicas de inyección de fallos y características deseables en herramientas que utilicen estas técnicas.

Los ingenieros utilizan la inyección de fallos para probar que tan seguros son los sistemas o componentes en el caso que suceda uno. La inyección de fallos prueba la detección de fallas, el aislamiento de fallas y las capacidades de reconfiguración y recuperación. Las técnicas de inyección permiten estimar cuan mal puede llegar a comportarse un sistema ante eventos externos, además es posible anticipar el comportamiento de un sistema en entornos operativos reales, y medir el impacto de estas fallas [1]. También se plantea estudiar los numerosos métodos, técnicas y finalmente se presentará una taxonomía con las técnicas de inyección existentes.

PLANTEAMIENTO DEL PROBLEMA

Debido al auge tecnológico muchas empresas y organizaciones a nivel mundial se ven en la obligación de implementar técnicas de inyección de fallos al proceso de desarrollo de sus sistemas debido a la necesidad de disponibilidad que estos demandan. Aunque se disponga de software perfecto, éste va a ser ejecutado en hardware que puede presentar fallas, tal vez con una tasa muy baja, pero fallas al fin. Estos fallos pueden conducir al software a estados no considerados y generar desviaciones en el comportamiento del

sistema, haciéndolo apartarse de sus objetivos operacionales. Las técnicas de inyección de fallos se clasifican desde varias perspectivas, como el tipo de sistema en el que es aplicada (un sistema real o prototipo), el tipo de fallos que es inyectado (físicos o software), o el mecanismo que ejecuta la acción de inyección (hardware o software), así mismo, existen nuevas técnicas de inyección que han surgido dado el avance y los requerimientos tecnológicos. [2].

Un caso en particular de aplicación de inyección de fallos se dio en una empresa de desarrollo de controladores de última generación altamente fiables para centrales hidroeléctricas. Por razones de rendimiento y económicas, el sistema de la empresa se basa en un procesador comercial OTS ARM926EJ-S. Esta CPU se diseñó y fabricó para aplicaciones multimedia sin grandes requisitos de fiabilidad, por lo que se optó por integrar el marco de inyección de fallos virtuales en la cadena de herramientas de desarrollo de software con el propósito de solventar los problemas de fiabilidad [3].

Es importante tener claro los conceptos de error, defecto y falla ya que estas terminologías serán muy utilizadas en el transcurso del estudio. Un defecto es una anomalía en el software, en el hardware o en los datos que tiene el potencial de causar errores y fallos. Los defectos son las causas de los errores, pero no todo defecto lleva a un error. Un fallo es la ocurrencia de la condición inválida o valor incorrecto en el sistema. Un fallo es la parte del estado del sistema, que es susceptible de ocasionar un error [4].

Existen muchas técnicas de inyección de fallos, pero no existe una clasificación actualizada con las más factibles para ser aplicadas y esto dependiendo del sistema en el que se requiera. Como consecuencia, habiendo determinado la problemática, surgen diferentes interrogantes; (i) ¿Cuáles son las técnicas de inyección de fallos existentes?, (ii) ¿Cuál es la técnica de inyección de fallos que brinda mejores niveles de fiabilidad?, (iii) ¿En qué tipos de sistemas se pueden aplicar técnicas de inyección de fallos?

En el transcurso del documento se detallan conceptualizaciones necesarias para el entendimiento de inyección de fallos, se profundiza en las distintas técnicas existentes y varias tecnologías utilizadas dependiendo del tipo de inyección, se exponen diferencias, que estas presentan y como cierre de este trabajo se presenta una taxonomía de las técnicas de inyección de fallos llevado a cabo por medio de un estudio documental.

JUSTIFICACIÓN

En la actualidad el que un sistema informático falle es motivo de preocupación, requiriendo atención urgente para plantear soluciones inmediatas. Es de suma importancia estar preparados, por tal motivo en esta investigación se detallan técnicas de inyección de fallos, específicamente se plantea presentar un catálogo con las principales o más utilizadas técnicas de inyección de fallos, las cuales permiten desarrollar sistemas muchos más confiables y con alta fiabilidad, permitiendo anticipar diferentes inconvenientes que se presenten en los sistemas. En este contexto es importante este trabajo investigativo ya que se refuerza la idea de utilizar estas técnicas para lograr tener una medida de calidad del software o hardware bajo situaciones externas que pueden ser predecibles pero que generalmente no son tomadas en cuenta. Se podrá identificar técnicas de inyección de fallos por medio de una taxonomía, lo cual facilita el análisis y la comprensión de estas, permitiendo identificar la que mejor convenga según el sistema que se la desee implementar.

OBJETIVOS

GENERAL

Realizar un estudio documental de las principales técnicas de inyección de fallos aplicables en sistemas distribuidos para establecer buenas prácticas a desarrolladores permitiéndoles crear sistemas con un mayor grado de fiabilidad.

ESPECÍFICOS

a) Identificar las principales técnicas, métodos y tecnologías de inyección de fallos aplicables en los sistemas distribuidos.
b) Comparar las técnicas de inyección de fallos en base a los enfoques cualitativos y cuantitativos según los parámetros de genericidad, accesibilidad y controlabilidad, neutralidad, automatización, precisión y coste.
c) Realizar una taxonomía de las técnicas de inyección de fallos que se puedan llevar a cabo en los sistemas distribuidos.

CAPITULO 1: MARCO TEÓRICO

1.1. BASES CONCEPTUALES

Para llevar a cabo este proyecto investigativo es necesario conocer diferentes conceptualizaciones fundamentales para una mejor comprensión de las temáticas sirviendo en el desarrollo del estudio.

1.1.1. Sistemas distribuidos

Sistema cuyos componentes de hardware o software situados en ordenadores en red, se comunican y establecen sus acciones únicamente mediante el paso de mensajes. Esta sencilla definición abarca toda la gama de sistemas en los que los ordenadores conectados en red pueden ser útiles [5]. Del mismo modo, [6] lo define como una colección de elementos informáticos autónomos que aparece ante sus usuarios como un único sistema coherente.

Hay varios aspectos importantes a resaltar en esta definición. Primeramente, se dice que un sistema distribuido está constituido de componentes (es decir, ordenadores) autónomos. Seguidamente, se tiene otro aspecto importante en el que los usuarios (personas o programas) deducen que interactúan con un único sistema. Esto significa que de una u otra manera sus componentes autónomos requieren colaborar entre sí [7]. A continuación, en la Figura *1* se ilustra un ejemplo de sistema distribuido.

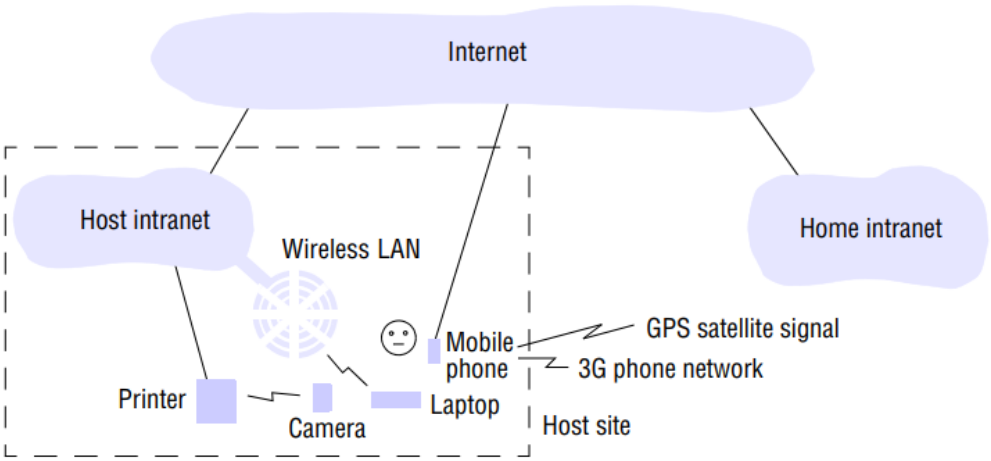

Figura 1. Conjunto de dispositivos en un sistema distribuido [5].

1.1.1.1 Principales características

- **Disponibilidad:** Los servicios ofrecidos por un sistema distribuido deberían estar siempre disponibles. Para ello, las aplicaciones deberán estar compuestas por múltiples módulos y cada uno de esos módulos debería replicarse de manera que cuando algún ordenador falle siempre haya otras copias del módulo en ordenadores que no fallen [8].
- **Ocultación:** Debido a la imagen de sistema único y coherente, se ocultan las diferencias existentes entre todos los ordenadores que componen el sistema. Además, tampoco resultará perceptible la complejidad de los mecanismos de comunicación que lleguen a ser necesarios para que las actividades ejecutadas en cada uno de estos ordenadores cooperen entre sí [8].
- **Acceso homogéneo:** Independientemente del sitio desde el que se acceda, todos reciben una misma interfaz. La interacción de los usuarios y las aplicaciones con el sistema se da de forma homogénea, con independencia del ordenador concreto empleado para atender tal acceso [8].
- **Escalabilidad:** Como el sistema ya está compuesto por múltiples ordenadores independientes no debería resultar difícil la incorporación de más ordenadores para asistir un número mayor de usuarios [8].

1.1.1.2 Tipos

En la literatura se describen tres tipos de sistemas distribuidos; sistemas distribuidos de cómputo, sistemas de información distribuidos y sistemas omnipresentes (que son naturalmente distribuidos).

Sistemas distribuidos de cómputo

Existes sistemas distribuidos muy importantes los cuales son utilizados para llevar a cabo cómputos de alto rendimiento. Es posible enfatizar entre dos subgrupos. En el clúster de computadoras, el hardware implícito consta de un grupo de estaciones de trabajo semejantes, u ordenadores personales, enlazadas cercanamente a través de una red local de alta velocidad. También, todos los nodos corren el mismo sistema operativo. En otro sentido, la computación en malla (grid) difiere mucho en comparación con la computación en clúster. La computación en malla consta de sistemas distribuidos creados

como un grupo de sistemas de cómputo, donde cada uno de los sistemas podrían caer en un dominio administrativo diferente, y podría ser muy diverso al referirnos del hardware, software, y la tecnología de red instalada [9].

Desde la perspectiva de la computación en red, el siguiente paso lógico es simplemente externalizar toda la infraestructura necesaria para las aplicaciones de cálculo intensivo. En esencia, en esto consiste la computación en nube que proporciona las facilidades para llevar a cabo dinámicamente una infraestructura y componer lo necesario a partir de los servicios que se disponen [6].

- Sistema de cómputo en grupo

  Estos sistemas se popularizaron debido a las mejoras precio-rendimiento de los computadores personales y equipos de trabajo. Por tal razón, se volvió factible tanto financiera como técnicamente construir supercomputadoras utilizando tecnología a través de la simple conexión de sencillas computadoras establecidas en una red de alta velocidad. En consecuencia, la computación en clúster se emplea en casos que se requiera de programación paralela, esto porque un solo programa (de cómputo intensivo) corre simultáneamente en múltiples computadoras [9]. En la Figura *2*Figura *2*, se presenta un ejemplo de sistema clúster de computadoras.

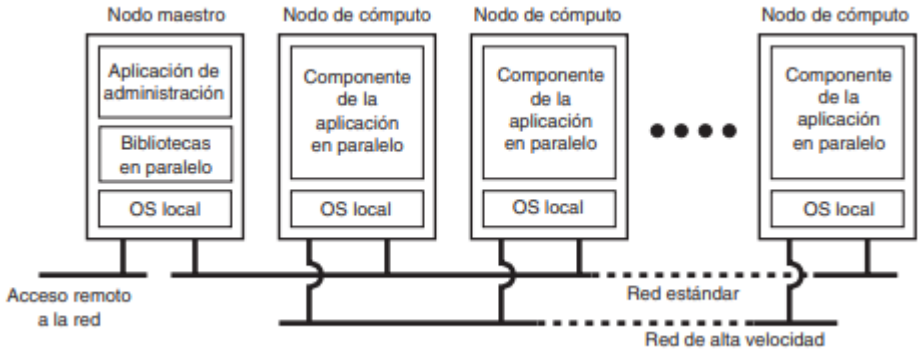

Figura 2. Ejemplo de sistema de computadoras en grupo [9].

- Sistema de cómputo en malla

  Un punto importante en un sistema de computación en malla es que los recursos de varias organizaciones se reúnen para llevar a cabo la colaboración de un conjunto de personas de diferentes instituciones, formando de hecho un conjunto de sistemas. Esta colaboración se lleva a cabo de manera de una organización virtual. Los procesos pertenecientes a la misma organización virtual pueden acceder a los recursos proporcionados a esa organización. Por lo general, los

recursos consisten en servidores de computación (incluidos los superordenadores, posiblemente implementados como ordenadores de clúster), instalaciones de almacenamiento y bases de datos. De igual manera, se pueden proporcionar dispositivos especiales en red, como sensores, telescopios, entre otros. En la Figura *3* se presenta una arquitectura de sistema de cómputo en malla [9].

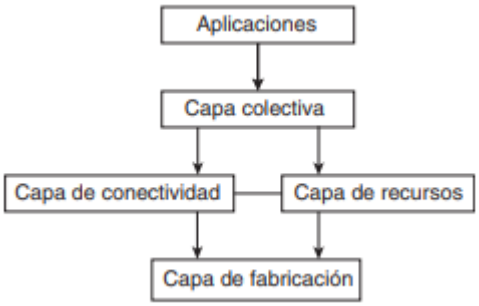

Figura 3. Arquitectura en capas para sistemas de cómputo en malla [9].

- Computación en la nube

  Se caracteriza por un conjunto de recursos virtualizados fácilmente utilizables y accesibles. En la Figura *4* se muestra un esquema de computación en la nube. Surgen varias interrogantes ¿Qué recursos son los que se utilizan? y ¿cómo se utilizan estos recursos? pueden configurarse dinámicamente, lo que significa el fundamento para la escalabilidad: si se requiere hacer más carga de trabajo, el usuario simplemente adquiere más recursos según sea necesario. El vínculo con la computación de servicios está formado por el hecho de que la computación en nube se basa según un modelo de subscripción o de pago por uso, en el que se brindan garantías a través acuerdos de nivel de servicios personalizados [9].

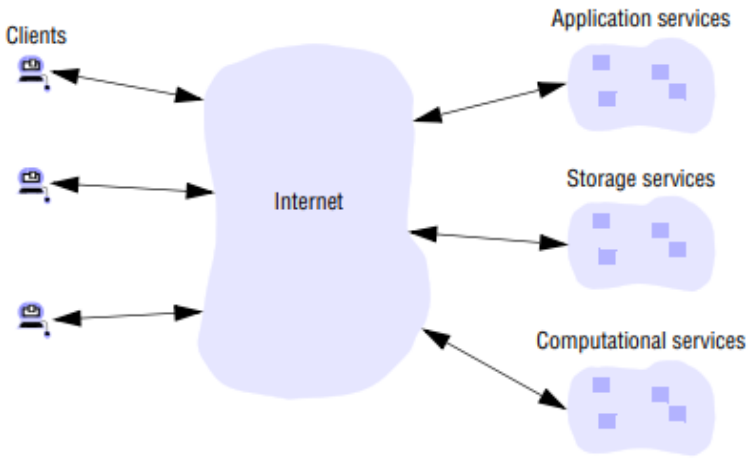

Figura 4. Esquema de computación en la nube [5].

Sistemas de información distribuidos

Debido a la sofisticación de las aplicaciones, éstas se iban separando de manera progresiva en componentes independientes resaltando los componentes de (base de datos y los de proceso), se clarificó que las integraciones deben asimismo tomar parte al permitir que las aplicaciones mantengan comunicación entre sí. Esto ha sido el resultante de grandes industrias que se dedican a integrar aplicaciones empresariales (por sus siglas en inglés, EAI). Por consiguiente, se enfocará en este par de formas de sistemas distribuidos[6], [9].

- Sistemas de procesamiento de transacciones

  Para catalogar esta explicación, se enfocará en las aplicaciones de bases de datos. Es común que las operaciones se realicen en forma de transacciones en las bases de datos. El hecho de programar empleando transacciones necesita primitivas de transacciones especiales facilitadas por el sistema distribuido implícito o por el lenguaje del sistema en tiempo de ejecución. [9].

- Integración de aplicaciones empresariales

  Dado el desligue progresivo de las aplicaciones y sus bases de datos, se hizo evidentemente necesaria la integración de aplicaciones independientes a sus bases de datos por las instalaciones. Por lo general, los componentes que conformaban una aplicación debían establecer comunicaciones entré sí directamente y no únicamente por medio de un patrón petición-respuesta soportado por sistemas de procesamiento de transacciones [9].

Sistemas omnipresentes o sistemas distribuidos masivos

Están pensados para integrarse de forma natural en el entorno. También son naturalmente sistemas distribuidos, y ciertamente cumplen caracterizaciones antes mencionadas. Lo que los hace únicos en comparación con los sistemas informáticos y de información descritos hasta ahora, es que la separación entre los usuarios y los componentes del sistema es mucho más difusa. A menudo no hay una única interfaz dedicada, como una combinación de pantalla y teclado. En cambio, un sistema omnipresente suele estar equipado con muchos sensores que recogen diversos aspectos del comportamiento del usuario. Asimismo, puede tener una miríada de actuadores que proporcionan información y retroalimentación, a menudo incluso con el objetivo de dirigir el comportamiento. Muchos dispositivos de los sistemas omnipresentes se caracterizan por su reducido

tamaño, funcionar con baterías, ser móviles y tener una única conexión inalámbrica, aunque no siempre estas características se aplican a la totalidad de dispositivos [6].

1.1.1.3 Taxonomía de los sistemas distribuidos

Los sistemas distribuidos se pueden clasificar de la siguiente manera basándose en su taxonomía:

- Sistemas con software en hardware tenuemente acoplados: sistema operativo de red, un sistema de archivo de red, del inglés (Network File System, NFS).
- Sistemas con software en hardware sólidamente acoplados: sistemas operativos de multiprocesador (sistemas paralelos).
- Sistemas con software sólidamente acoplados en hardware tenuemente acoplados: sistemas verdaderamente distribuidos (imagen de sistema único).

Los sistemas operativos de red son casos particulares de sistemas distribuidos con software y hardware tenuemente acoplados. Varios de los servicios de estos sistemas tienes son: conexión remota entre ordenadores, copia remota de archivos entre diferentes computadoras, sistema de archivos global compartidos [10].

1.1.1.4 Ejemplos de los sistemas distribuidos

Estos ejemplos están apoyados en redes de ordenadores populares y muy utilizados como la internet, siendo un amplio grupo de redes de computadoras de varios tipos interconectadas, donde los programas son ejecutados interactuando mediante el paso de mensajes, idénticamente, la intranet, la cual es una porción de la internet pero que tiene una limitante que se puede configurar para permitir el cumplimiento de políticas de área local, por otro lado, en el ámbito de la computación móvil y ubicua donde las tendencias tecnológicas de miniaturización de dispositivos y redes inalámbricas han permitido y facilitado la integración de dispositivos de ordenadores pequeños en sistemas distribuidos como: ordenadores portátiles, asistentes personales, teléfonos móviles entre otros. A continuación, en la Tabla *1* se describe una serie de sectores de aplicación comerciales o sociales clave, destacando algunos de los usos asociados, establecidos o emergentes, de la tecnología de sistemas distribuidos [5], [11].

Tabla 1. Dominios de aplicación seleccionados y aplicaciones en red asociadas [5].

Finanzas y comercio	El crecimiento del comercio electrónico, ejemplificado por compañías como Amazon y eBay, y las tecnologías de pago subyacentes, como PayPal; la aparición asociada de la banca y el comercio en línea y también los complejos sistemas de difusión de información para los mercados financieros [5].
La sociedad de la información	El desarrollo de la World Wide Web como almacén de información y conocimiento; el desarrollo y crecimiento de motores de búsqueda en la red como Google y Bing, Yahoo para buscar en este vasto repositorio; la aparición de bibliotecas digitales y la digitalización a gran escala de fuentes de información heredadas como los libros (por ejemplo, Google Books); la creciente importancia de los contenidos generados por los usuarios a través de sitios como YouTube, Wikipedia y Flickr; la aparición de las redes sociales a través de servicios como Facebook y MySpace [5].
Industrias creativas y entretenimiento	La aparición de los juegos en línea como forma novedosa y altamente interactiva de entretenimiento; la disponibilidad de música y películas en el hogar a través de los centros multimedia en red y más ampliamente en Internet a través de contenidos descargables o en streaming; la influencia e impacto de los contenidos generados por el usuario (como se ha mencionado anteriormente) como una nueva forma de creatividad, por ejemplo a través de servicios como YouTube; la creación de nuevas formas de arte y entretenimiento posibilitadas por las tecnologías emergentes (incluidas las de red) [5].
Cuidado de la salud	El crecimiento de la informática de la salud como disciplina con su énfasis en los registros electrónicos de pacientes en línea y cuestiones relacionadas con la privacidad; el papel cada vez más importante de la telemedicina en el apoyo al diagnóstico remoto o servicios más avanzados como la cirugía a distancia (incluido el trabajo colaborativo entre equipos de atención médica); la aplicación cada vez mayor de la tecnología de redes y sistemas integrados en la vida asistida, por ejemplo, para controlar a las personas mayores en sus propios hogares [5].

Educación	La aparición del aprendizaje electrónico a través de, por ejemplo, herramientas basadas en la web como los entornos virtuales de aprendizaje; el apoyo asociado al aprendizaje a distancia; el apoyo al aprendizaje colaborativo o basado en la comunidad [5].
Transporte y logística	La utilización de tecnologías de localización como el GPS en los sistemas de búsqueda de rutas y en los sistemas más generales de gestión del tráfico; el propio coche moderno como ejemplo de un sistema distribuido complejo (también se aplica a otras maneras para transportarse como los aviones); el desarrollo de servicios de mapas basados en la web como MapQuest, Google Maps y Google Earth [5].
Ciencia	La aparición de la Grid como fundamental tecnología para la eScience, incluyendo el uso de complejas redes de ordenadores para apoyar el almacenamiento, análisis y procesamiento de (a menudo muy grandes cantidades de) datos científicos; el uso asociado de la Grid como tecnología facilitadora de la colaboración mundial entre grupos de científicos [5].
Gestion Ambiental	El uso de tecnología de sensores (en red) para monitorear y gestionar el medio ambiente, por ejemplo, para proporcionar una alerta temprana de desastres naturales como terremotos, inundaciones o tsunamis y para coordinar la respuesta de emergencia; la recopilación y el análisis de parámetros ambientales globales para comprender mejor fenómenos naturales complejos como el cambio climático [5].

1.1.2 Confiabilidad y fiabilidad

Con formalidad [12], la confiabilidad se establece como la probabilidad de que un sistema opere como es esperado en un intervalo de tiempo especificado. Del mismo modo [1], la define como la propiedad de un sistema tal que se puede confiar en el servicio que ofrece.

La confiabilidad es un concepto muy general que comprende varios atributos; disponibilidad: correcto acceso a un servicio, fiabilidad: continuidad de un servicio, seguridad: ausencia de consecuencias catastróficas, integridad: ausencia de alteraciones inadecuadas, mantenimiento: capacidad para soportar modificaciones y reparaciones.

1.1.3 Inyección de fallos

Los sistemas fallan de forma compleja e inesperada, debido a combinaciones imprevistas de eventos e interacciones entre componentes de hardware y software, por esta razón la inyección de fallos se presenta como solución a esa problemática permitiendo medir la robustez de los sistemas como la latencia, la propagación y la cobertura de los fallos [13], [14].

La inyección de fallos es una técnica muy conocida, potente y útil para evaluar la fiabilidad y observar el impacto de los fallos generados en el comportamiento del sistema [15]. Se basa en la ejecución de experimentos monitoreados y debe ser acompañada de un análisis previo, en el cual se determinan los fallos a introducir, los distintos estados y puntos en los cuales se deben inyectar dichos fallos.

Después de dicho análisis se procede al diseño de los casos de prueba, especificando los tipos de fallos a introducir, puntos y estados de prueba y resultados esperados [16], [17]. Existen cinco categorías principales de enfoques de inyección de fallos: inyección de fallos por hardware (WHIFI), inyección de fallos por emulación (EFI), inyección de fallos por software (SWIFI), inyección de fallos por simulación (SFI) e inyección de fallos híbridas, con diferentes ventajas y dificultades. La inyección de fallos de hardware utiliza fuentes físicas externas para introducir fallos en el hardware del sistema. Es el método más rápido, pero también el más costoso, ya que requiere un hardware especial. Además, sólo puede aplicarse cuando la plataforma de hardware está disponible, y presenta un alto riesgo de dañar el sistema inyectado. La inyección de fallos basada en la emulación se basa en el uso de una matriz de puertas lógicas programable en campo, del inglés (Field Programmable Gate Arrays, FPGAs) con el objetivo de proporcionar un rendimiento similar a los enfoques HFI sin su alto coste y riesgo de daños. Sin embargo, la EFI se limita clásicamente a la inyección de fallos de tipo stuck-at y requiere un modelo sintetizable del sistema sometido a evaluación. El análisis temprano sin necesidad de HW de propósito especial puede realizarse mediante el uso de herramientas SWIFI y SFI. SWIFI se basa en la alteración del software que se ejecuta en el sistema bajo análisis. Los experimentos de SWIFI pueden llevarse a cabo casi en tiempo real, pero el software bajo prueba debe estar instrumentado y los fallos no pueden inyectarse en lugares inaccesibles para el software [18].

1.1.4 Modelo de fallos

Existe una gran variedad de enfoques para razonar sobre un modelo de fallos que describe cómo se ve afectado un sistema por la inyección de fallos. El objetivo de este modelo es describir cómo el sistema se ve afectado por el ataque de inyección de fallos de manera que pueda utilizarse para entender o razonar sobre ese fallo. En este caso, la atención se centra en cómo se ve afectado el sistema, y en describirlo de manera que pueda ser simulado y comprobado mediante las metodologías experimentales consideradas. Esto significa que los modelos de fallos considerados aquí suelen estar relacionados con modificaciones del programa binario, por ejemplo, cambiando el valor de un bit o byte dentro del propio programa [19].

1.1.5 Técnicas

1.1.5.1 Inyección de fallos basada en Hardware (HWIFI)

La inyección de fallos basada en hardware, del inglés (hardware implemented fault injection, HWIFI) utiliza fuentes físicas externas para introducir fallos en el hardware del sistema. Esto significa que el fallo se inyecta en el sistema objetivo real, por ejemplo, cambiar el valor de un pin, o perturbar el hardware con parámetros del entorno como la radiación de iones pesados y las interferencias electromagnéticas [15]. También puede hacerse inyectando caídas de tensión en los carriles de alimentación del hardware para simular las perturbaciones de la fuente de alimentación, y mediante rayos láser [20]. Estas técnicas permiten acceder a algunas ubicaciones a las que es difícil acceder mediante técnicas basadas en software. Sin embargo, son caras en términos de tiempo de ejecución y coste de hardware [17].

Hay dos grupos principales de inyección de fallos físicos: inyección con contacto y sin contacto.

- Contacto: los fallos se inyectan externamente en el nivel de los pines, dicho de otro modo, el inyector y el sistema objetivo está directamente en contacto físico. Por ejemplo, produciendo cambios de voltaje.
- Sin contacto: no tiene contacto directo con el sistema de objetivo, los fallos se inyectan internamente con radiación de iones pesados o interferencia electromagnética, causando falsas corrientes dentro del chip objetivo [1].

En la inyección de fallos a nivel de pin, los fallos se inyectan forzando pull ups o pull downs en los pines de un CI. Esta técnica es limitada debido a que no todos los pines son

accesibles en los CIs modernos, sin embargo, la reproducibilidad y la capacidad de control son muy altas. En la inyección de iones pesados, el CI estudiado se bombardea con isótopos pesados cargados eléctricamente. Otra técnica es la inyección de fallos por interferencia electromagnética, que utiliza generadores de inducción como medio de inyección de fallos. La intrusividad de estas técnicas es bastante baja, pero la principal preocupación está relacionada con la insignificante capacidad de control de un lugar concreto, por lo que la reproducibilidad es especialmente difícil de conseguir. La inyección de fallos en modo de depuración en segundo plano (BDM) aprovecha la capacidad que ofrecen los microprocesadores modernos para probar y depurar el sistema. Debido al uso de las capacidades integradas en el chip, la intrusión de estas técnicas es significativa, y la reproducibilidad es alta [14].

La ventaja de las técnicas de inyección de fallos por hardware es la capacidad de acceder a algunos lugares a los que no es fácil acceder con otras técnicas. También son adecuadas para los sistemas que requieren una alta resolución temporal para la activación y la supervisión del hardware. Las desventajas de estas técnicas es que el enfoque de inyección necesita un hardware especial y requiere la accesibilidad al hardware del sistema objetivo, lo que a veces puede no ser fácil o ser muy caro de conseguir. Además, estas técnicas pueden presentar un alto riesgo de dañar el sistema estudiado [21].

1.1.5.2 Inyección de fallos basada en Software (SWIFI)

En la inyección de fallos implementada en software, del inglés (Software Implemented Fault Injection, SWIFI), los fallos inyectados simulan los que pueden producirse debido a los fallos del hardware, se realiza bien en tiempo de compilación, insertando los efectos de los fallos de hardware en el objetivo, o bien en tiempo de ejecución, utilizando tiempos de espera, excepciones, inserción de código o alterando el estado del objetivo para provocar los fallos, así mismo, estimulando las condiciones que tendría un fallo físico en el software y los datos, generando fallo en los registros de la CPU y/o en los elementos de memoria [16] [20].

Las técnicas SWIFI suelen implementarse generando cambios de bits en la memoria, emulación de segmentos de datos o instrucciones corruptos en el software bajo prueba. Estas técnicas suelen variar en relación con el propio método de inyección, utilizando banderas que detienen la ejecución del programa para realizar una inyección, o explotando llamadas al sistema [14]. No obstante, se pueden utilizar para aplicaciones y

sistemas operativos, lo cual es difícil de hacer con inyección de fallos por hardware. Si el objetivo es una aplicación, el inyector puede ser insertado en la propia aplicación, o entre las capas de la aplicación y el sistema operativo. Si el objetivo es el sistema operativo, el inyector debe ser incorporado en el sistema operativo. Aunque los métodos por software son flexibles, tiene sus deficiencias:

- No puede inyectar fallos en los lugares que son inaccesibles al software.
- El software necesario para el funcionamiento del inyector puede perturbar el trabajo que se ejecuta en el sistema objetivo, e incluso cambiar la estructura de software original. Un diseño cuidadoso del ambiente de inyección puede reducir al mínimo las alteraciones.
- Un alto tiempo de la resolución puede causar problemas. Para fallos de tiempo de latencia grande, tales como fallos de memoria, el tiempo de resolución puede no ser un problema. Para fallos de corta latencia, como las de bus y los fallos de CPU, la técnica puede fallar en la captura ciertos fallos de comportamiento, como la propagación [16].

La inyección de fallos basada en software también presenta un cuello de botella para las aplicaciones de ingeniería en términos de controlabilidad y operabilidad. Estudios recientes han revelado que no son precisas para analizar los fallos blandos que se originan en los flip-flops. Sin embargo, aún no se ha estudiado la eficacia de estas técnicas para evaluar todo el procesador, incluidos los archivos de registro y las matrices de caché [22], [23].

1.1.5.3 Inyección de fallos basada en Simulación (SFI)

En la inyección de fallos basada en la simulación, el sistema objetivo, así como los posibles fallos de hardware, se modelan y simulan mediante un programa de software. La simulación de fallos se realiza modificando el modelo de hardware o el estado del software del sistema objetivo. Esto significa que el sistema podría comportarse como si hubiera un fallo de hardware. Idénticamente [1], expresa que la inyección de fallos basada en simulación consiste en la construcción de un modelo del sistema bajo análisis, incluyendo un detallado modelo de simulación del procesador en uso. Esto significa que los errores o fallos de la simulación del sistema se producen de acuerdo con una distribución predeterminada. Los modelos de simulación son desarrollados generalmente utilizando lenguajes de descripción de hardware como por ejemplo el VHDL o Verilog.

El modelo debe ser una representación exacta de la realidad del sistema bajo análisis. En el periodo de delineación de un sistema, la simulación es un recurso de prueba de suma importancia para llevar a cabo una primera evaluación de las especificaciones y la confiabilidad. Como consecuencia, cuanto mayor sea el tiempo que tome detectar y resolver las dificultades de delineación que puedan encontrarse, más alto resultará en duración y económicamente. Contrastándola con el modelado analítico, la simulación muestra la posibilidad de poder modelar sistemas más complejos con un mayor índice de fiabilidad, sin limitarse a asunciones efectuadas para lograr que el modelo analítico sea matemáticamente accesible. La ventaja de las técnicas de inyección de fallos basadas en la simulación es que no hay riesgo de dañar el sistema en uso, no son intrusivas, es decir no hay necesidad de modificar el código fuente del diseño. Además, son más baratas en términos de tiempo y esfuerzo que las técnicas de hardware. También tienen una mayor capacidad de control y reconocimiento del actuar del sistema con la aparición de fallos [15]. Sin embargo, las técnicas de inyección de fallos basadas en la simulación pueden carecer de la precisión del modelo de fallos y del modelo del sistema. Por otra parte, tienen una pobre resolución temporal, lo que puede causar problemas de fidelidad, el rendimiento que se puede esperar en la práctica es del orden de unos pocos fallos por segundo o menos, dependiendo del esfuerzo de simulación requerido, esto puede ser insuficiente en muchos casos, se necesita un mayor esfuerzo para llevar a cabo el modelo, esto implica grandes costos y larga duración. La exactitud de los resultados depende de la fidelidad del modelo utilizado. Los modelos no necesariamente incluyen la totalidad de los defectos de diseño que se presenten en el verdadero hardware [1], [24].

Los simuladores de sistemas que simulan el comportamiento de un dispositivo con la precisión de un ciclo de reloj. Pueden estar basados en la ejecución, cuando se ejecuta directamente, o en la traza, cuando la simulación se realiza utilizando una traza de ejecución previamente generada.

Inyección de fallos en simuladores basados en la ejecución: En este tipo de simuladores, se integra un módulo de inyección en el diseño objetivo. El módulo de inyección de fallos puede integrarse como un módulo dedicado llamado saboteador. Está inactivo durante el funcionamiento normal y puede alterar el valor o la temporización cuando está activo. Los saboteadores pueden insertarse en serie o en paralelo al diseño objetivo. La inserción en serie, en su forma más sencilla, consiste en romper la ruta de la señal entre un conductor (salida) y su correspondiente receptor (entrada) y colocar un saboteador entre ambos. En su forma más compleja, es posible romper las rutas de señal

entre un conjunto de controladores y su correspondiente conjunto de receptores e insertar un saboteador. En el caso de la inserción en paralelo, un saboteador se añade simplemente como un conductor adicional para una señal resuelta. El otro enfoque de inyección de fallos es por medio la utilización de mutantes que se insertan modificando partes de los componentes del circuito objetivo. Estos dos enfoques presentan la ventaja de tolerar completamente los niveles de abstracción del sistema: eléctrico, lógico, funcional y arquitectónico. Estos enfoques permiten la reproducción completa de: cambios de un solo bit, alteraciones de bits seleccionados, corrupción de datos, reconexión de circuitos, alteración del reloj y efectos de intercambio de instrucciones [25].

Inyección de fallos en simuladores basados en trazas: Un ejemplo de herramientas de inyección de fallos que explotan las simulaciones basadas en trazas es el de Miele. Esta herramienta analiza la fiabilidad a nivel de sistema de los sistemas embebidos. El flujo de trabajo se organiza en tres fases principales: caracterización preliminar del sistema, configuración de la campaña experimental, y ejecución de la campaña experimental y post-procesamiento de los resultados. El diseñador especifica las acciones de monitorización y clasificación a nivel de aplicación y arquitectura. El mecanismo de depuración permite analizar la propagación de los fallos en varias funcionalidades de la aplicación ejecutada. El enfoque propuesto es muy adecuado para reproducir los efectos en la simulación de los cambios de un solo bit, las alteraciones de bits seleccionados, las corrupciones de datos y los intercambios de instrucciones. La principal ventaja de utilizar un simulador basado en trazas es la posibilidad de alterar partes específicas del sistema sin necesidad de alterar la estructura principal del mismo [25].

Existen varias herramientas para llevar a cabo inyecciones de basadas en simulación: MEFISTO [14], implementan los fallos mediante componentes VHDL que llevan a cabo la inyección de fallos (denominados saboteadores) y componentes modificados (corruptos) que sustituyen a los componentes libres de fallos en el sistema (denominados mutantes).

1.1.5.4 Inyección de fallos basada en Emulación (EFI)

La inyección de fallos basada en la emulación se ha introducido como una mejor solución para reducir el tiempo de ejecución en comparación con SFI. Suele basarse en el uso de una matriz de puertas lógicas programable en campo, del inglés (Field Programmable Gate Arrays, FPGAs) para acelerar la simulación de los fallos y explota los FPGAs para la emulación efectiva de los circuitos [15].

SFI es muy flexible, pero también bastante lenta. Para clasificar el efecto de cada uno de los fallos, hay que simular el circuito para un banco de pruebas completo. Las grandes campañas de inyección de fallos pueden requerir un enorme esfuerzo de simulación que no se puede permitir en la práctica. La solución para las grandes campañas de inyección de fallos es mejorar la velocidad emulando en una FPGA. La principal ventaja es que el circuito puede funcionar a velocidad. Por tanto, el diseño puede probarse en grandes bancos de pruebas, e incluso puede probarse la interacción con entornos reales (por ejemplo, dispositivos externos).

Dado que la tecnología de la FPGA puede ser diferente de la tecnología final del ASIC (circuitos complejos, como microprocesadores o circuitos integrados de aplicación específica), los detalles de bajo nivel del diseño (a nivel de transistores o de lógica) son diferentes. Sin embargo, la implementación de la FPGA muestra el mismo comportamiento en la RTL (nivel de transferencia de registros) y más allá. Desde este punto de vista, la inyección de fallos puede soportarse adecuadamente siempre que el modelo de fallos pueda describirse en la RTL [26].

1.1.5.5 Inyección de fallos Híbrida

El enfoque híbrido integra varias técnicas FI para un mejor análisis del sistema. Por ejemplo, la ejecución de inyección de fallos basada en hardware o software pueden proporcionar el beneficio significativo en términos de tiempo para trabajar con experimentaciones de inyección de fallos, se puede reducir tiempo inicial de la instalación. El enfoque híbrido puede combinar la versatilidad de la inyección de fallos por software, y la exactitud del monitoreo de hardware; esto es muy adecuado para medir latencias extremadamente cortas. Dada la significativa ganancia en la controlabilidad y observabilidad con un enfoque basado en la simulación, podría ser útil combinar un enfoque basado en la simulación con uno de los otros a fin de ejercer plenamente el sistema bajo análisis. Por ejemplo, la mayoría de los investigadores y los profesionales pueden optar por modelar una parte del sistema bajo análisis, como la Unidad de Aritmética y Lógica (ALU) en el microprocesador, en un nivel muy detallado, y realizar inyección basadas en simulación por el hecho de que el interior de los nodos de una ALU no es accesible mediante otro enfoque [1]. Existen varias herramientas para llevar a cabo inyecciones de fallos híbrida:

En [27] se propone la herramienta ASPHALT, la cual emplea metodología híbrida que permite llevar a cabo una pronta inyección en el hardware del sistema con fallos

transitorios. Se basa en combinar la técnica SWIFI con la simulación a nivel RT de un modelo en Verilog. En este trabajo se estudia el microprocesador ROMP del inglés (Risc-Oriented MicroProcessor, IBM) siendo un procesador de tipo RISC con morfología pipeline. Ciertamente, la rapidez es garantizada por la utilización de la técnica SWIFI, al efectuarse en el sistema en concreto o real, sin embargo, la simulación RT aumenta el parámetro de exactitud de los modelos de fallos. Varios de los modelos utilizados para el proceso de simulación son: carga de registro ejecutada erróneamente, carga de registro sin efectuar, cambio del contenido de registros o también conocido como (bit-flip), carga todo a '0' o '1'. La autenticidad de los modelos se evidencia mediante la comparación entre la simulación RT y la simulación a nivel lógico, esto mediante fallos stuck-at transitorios. Finalmente, se detalla que el 97% de fallos a nivel lógico los cubren los fallos RT.

En [27] se plantea una herramienta con dos enfoques que se basa en la técnica SWFI con la basada en simulación para agilizar la inyección (algo parecido a ASPHALT), por lo que el modelo se lleva a cabo en el sistema. Con INERTE del inglés (Integrated NExus-based Real-Time fault injection tool for Embedded systems) es posible inyectar fallos transitorios en memoria sin insertar alguna sobrecarga temporal. El seguimiento o monitoreo del sistema tras el proceso de inyección se efectúa empleando un dispositivo hardware, por lo común un emulador ya sea de microprocesador o del microcontrolador sujeto a estudio. Esta herramienta se presenta como híbrida [27].

1.2. ANTECEDENTES

A continuación, se presentan los análisis y las descripciones de investigaciones que se realizaron llevando a cabo una exploración de la literatura en base al tema en cuestión. En principio en el estudio titulado ‘‘Validación por inyección de fallos en VHDL de la arquitectura TTA’’ utilizaron la inyección de fallos basada en simulación para estudiar más en detalle los fenómenos observados en experimentos de inyección de fallos mediante iones pesados (HWIFI). Compararon la inyección de fallos basada en simulación con la inyección de fallos implementada mediante Scan-Chain, e hicieron una extensa comparación de diferentes técnicas de inyección de fallos física [28]. En otro sentido, en [29] presentaron un estudio donde se evalúa un sistema empotrado distribuido de control utilizando la inyección de fallos implementada por software y la inyección de fallos a nivel de pin, y en el que se muestra la complementariedad de ambas técnicas. Combinaron tres técnicas de inyección de fallos (inyección de fallos a nivel de pin,

inyección de fallos implementada por software e inyección de fallos mediante iones pesados) para evaluar el protocolo TTP/C.

Considerar también el estudio titulado ‘‘Fault Injection Analytics: A Novel Approach to Discover Failure Modes in Cloud-Computing Systems’’. Los autores expresaron que los experimentos de FI producen cantidades masivas de datos, y el análisis manual de estos datos es ineficiente y propenso a fallos, ya que un analista en particular puede pasar por alto modos de fallos severos que aún se desconocen, por tal motivo [13], introdujeron un nuevo paradigma de análisis de inyección de fallos pero aplicando el aprendizaje mecánico no supervisado en los rastros de ejecución del sistema inyectado, para facilitar el descubrimiento e interpretación de los modos de falla. Evaluaron el enfoque propuesto en el contexto de los experimentos de inyección de fallas en la plataforma de computación en nube OpenStack, llegando a la conclusión que el enfoque puede identificar con precisión los modos de falla con un bajo costo de computación.

A continuación, el estudio titulado ‘‘Fault injection for FPGA applications in the space’’. Los autores [20], realizaron la inyección de fallos para medir la robustez de su sistema, detallaron que tan tolerante es a los fallos durante el proceso de diseño, también determinaron los parámetros de fiabilidad del inyectar fallos tanto permanentes como transitorias a diferentes tasas de falla que varían con el tiempo en las órbitas espaciales. Sus resultados obtenidos de la simulación ilustraron que las variaciones calculadas se aproximan a las tasas reales en órbita. Este estudio fue de mucha ayuda ya que facilita la comprensión de la inyección de fallos basada en hardware.

Otro estudio cuyo título es ‘‘KITO tool: A fault injection environment in Linux kernel data structures’’. Ellos [14], presentaron una herramienta específica para inyectar fallos, denominada KITO, para evaluar los efectos de las fallas en la memoria que contiene estructuras de datos pertenecientes a un sistema operativo basado en Unix. Realizaron un análisis experimental de un gran conjunto de elementos de memoria del propio Sistema Operativo, mientras que el sistema estaba sujeto a la presión de programas de referencia que utilizaban diferentes elementos del núcleo de Linux. Los resultados experimentales que obtuvieron mostraron que los aspectos de sincronización del núcleo son susceptibles de un conjunto significativo de posibles fallos que van desde la degradación del rendimiento hasta el fracaso en completar con éxito la aplicación de referencia.

Por consiguiente, el estudio titulado ‘‘FIJI: Fault InJection Instrumenter’’. En esta investigación los autores [29], resumieron brevemente diversos métodos de inyección de fallos, centrándose en los métodos que son capaces de someter a tensión las redes críticas de un diseño que se ejecuta en un equipo real sin necesidad de volver a sintetizar. Si bien las herramientas de última generación pueden funcionar con diseños complejos, a menudo carecen de control sobre el momento exacto en que se producen los eventos de inyección (lo que es importante para seguir la respuesta del sistema ante los fallos en una simulación lógica) y/o utilizan una gran cantidad de recursos de FPGA. Para superar esos problemas, presentaron un marco de inyección de fallas basado en una lista de recursos que ahorra recursos. Instrumento de inyección de fallas (FIJI) que puede apuntar a redes individuales en tiempo de ejecución de pruebas. El marco de trabajo de la FIJI fue puesto a disposición del público por los autores bajo una licencia de código abierto.

Finalmente, es estudio titulado ‘‘A Virtual Fault Injection Framework for Reliability-Aware Software Development’’. Los autores [3], describieron que para desarrollar de manera eficiente software tolerante a fallas, se necesita la inyección de fallos en las fases iniciales de desarrollo. Sin embargo, los enfoques comunes requieren productos fabricados o modelos de hardware detallados. Por lo tanto, estas técnicas normalmente no son aplicables si los proveedores de software y hardware son proveedores independientes. Además, el aumento de componentes de software OTS de terceros limita los medios para inyectar fallos. Ellos, presentaron un marco de inyección de fallos virtuales que simula modelos de fallas alineados con estándares de seguridad y admite componentes de software OTS, así como procesadores integrados ampliamente utilizados, como núcleos ARM. Además, mostraron cómo integrar el marco en varias etapas de desarrollo de software. Por último, ilustraron la viabilidad del enfoque ejemplificando la integración del marco en el desarrollo de un sistema crítico para la seguridad industrial.

1.3. FUNDAMENTACIÓN LEGAL

En la Ley de Propiedad Intelectual en el Libro I, en su Título I.- De los derechos de autor y derechos conexos, Capítulo I.- Del derecho de autor, Sección I Preceptos generales, el Art. 4 menciona que “Se reconocen y garantizan los derechos de los autores y los derechos de los demás titulares sobre sus obras” [30]. En consecuencia, de lo ya mencionado, el estudio investigativo cumplirá con todos los parámetros de autoría, respetando los

lineamientos a fines del derecho de autor, por consiguiente, su construcción, implementación y desarrollo.

Además, de acuerdo con la Ley Orgánica de educación superior en el Título I de Ámbito, objeto, fines y principios del sistema de educación superior, Capítulo 2.- Fines de la educación superior, en el Art. 8.- Fines de la Educación Superior, describe en uno de sus apartados que "Fomentar y ejecutar programas de investigación de carácter científico, tecnológico y pedagógico que coadyuven al mejoramiento y protección del ambiente" [31]. Este estudio, tiene la intención de proporcionar un catálogo con las mejores técnicas FI, que resuelve la problemática de crear sistemas no tan propensos a fallos, fomentando la investigación tecnológica en virtud de colaborar en el desarrollo socioambiental de los ciudadanos.

De acuerdo con la Constitución de la Republica del Ecuador en el Titulo VII de Régimen de buen vivir, Capitulo Primero de Inclusión y equidad, Sección octava de Ciencia, tecnología, innovación y saberes ancestrales, que en el Art. 385, menciona que "Desarrollar tecnologías e innovaciones que impulsen la producción nacional, eleven la eficiencia y productividad, mejoren la calidad de vida y contribuyan a la realización del buen vivir"[32]. Este trabajo investigativo favorece a la comunidad de tal manera que proporciona buenas prácticas para crear sistemas mucho más estables.

CAPITULO 2: MATERIALES Y MÉTODOS

2.1 Delimitación de la investigación

El trabajo investigativo fue delimitado espacialmente según el concepto de ‘‘inyección de fallos basadas en hardware’’, el cual está comprendido dentro de la temática ‘‘sistemas distribuidos’’ del tema en cuestión. En vista que el contexto de la investigación es global se utilizaron bases de datos internacionales como: SCOPUS y EBSCOHost y bibliotecas digitales como IEEE Xplore y ACM. Por otro lado, la investigación se llevó a cabo en base a una delimitación temporal entre los años (2015-2021) para la recolección de la información de las fuentes antes mencionadas.

2.2 Tipos de investigación

De acuerdo con el nivel de profundidad de la investigación, este estudio es del tipo documental, debido a que la información utilizada para la investigación fue extraída de bases de datos bibliográficas para su posterior análisis. Agregando a lo anterior, la

investigación se estableció en base a un paradigma metodológico mixto, esto porque se fundamentó en los enfoques: cualitativo y cuantitativo. Puesto que, los parámetros para la realización de la taxonomía de las mejores técnicas de inyección de fallos fueron; genericidad, accesibilidad, controlabilidad, neutralidad, automatización, precisión y coste, siendo unos cualitativos y otros cuantitativos. Para un mejor entendimiento la investigación cualitativa se define como: ‘‘aquella que utiliza preferente o exclusivamente información medible y cuyo análisis se dirige a lograr descripciones detalladas de los fenómenos estudiados’’[33]. La investigación cuantitativa ‘‘utiliza la recolección de datos para probar hipótesis con base en la medición numérica y el análisis estadístico, con el fin establecer pautas de comportamiento y probar teorías’’ [34].

2.3 Métodos de la investigación

Para llevar a cabo el trabajo investigativo se emplearon los siguientes métodos: deductivo e inductivo. El método deductivo se empleó para la obtención de información en las diferentes bases de datos antes mencionadas, con el objetivo de amplificar los conocimientos sobre el tema. De igual manera, se empleó el método inductivo, esto porque partiendo de diferentes artículos y publicaciones referente a las inyecciones de fallos se podrá establecer las mejores técnicas por medio de una taxonomía.

2.4 Variables e indicadores

Las variables del trabajo de investigación e indicadores se presentan en la Tabla *2*. Las variables fueron: técnicas de inyección y tipos de sistemas, estas variables cuentan con (7 y 3) indicadores respectivamente tanto cuantitativos como cualitativos.

Tabla 2. Variables e indicadores sujetos a estudio.

			Operacionalización	
Variables	**Definición**	**Dimensiones**	**Indicadores**	**Tipo de variable**
Técnicas de inyección	Modo en cómo se realiza el proceso de inyección en un sistema determinado.	Tipos/clasificación	genericidad	Cualitativo
			accesibilidad	Cualitativo
			controlabilidad	Cualitativo
			neutralidad	Cuantitativo
			automatización	Cualitativo
			precisión	Cuantitativo
			costo	Cuantitativo

Seguidamente, se describen los indicadores de las variables listadas en la Tabla 2:

Indicadores de la variable (Tipos/clasificación**)**

- Genericidad: independencia de la técnica con relación del sistema.
- Accesibilidad y controlabilidad: capacidad de acceso a los puntos de inyección, y la facultad de control de los atributos de los fallos inyectados.
- Neutralidad: medida de importancia de los cambios generados por la técnica de inyección sobre el sistema.
- Automatización.
- Precisión: evaluación de adaptación de la técnica a los supuestos fallos considerados en los objetivos de la campaña.
- Costo: valor de implementación de la técnica.

2.5 Técnicas e instrumentos de recolección de datos

Para la recolección de datos se aplicó la técnica de investigación documental o bibliográfica, que permitió identificar estudios en los que se ha aplicado técnicas de inyección de fallos, agregando a lo anterior, se aplicó la metodología de mapeo sistemático, cuyos pasos esenciales del proceso son: definición de las preguntas de investigación, búsqueda de artículos relevantes, inclusión y exclusión de artículos, búsqueda de palabras claves y la extracción y el mapeo de los datos. Cada paso del proceso tiene un resultado en consecuencia se obtiene es el mapa sistemático [35].

a) *Definición de las preguntas de investigación.* Establecer preguntas referentes al tema en estudio. Proporciona una visión generalizada de un área de investigación e identifica la cantidad, el tipo de investigación y los resultados encontrados dentro de ella [35].
b) *Búsqueda de artículos relevantes*. Realizar la búsqueda de los estudios en bases de datos científicas, utilizando cadenas con términos claves del tema en estudio, permitiendo capturas información relevante [35].
c) *Inclusión y exclusión de artículos.* Utilizar criterios de inclusión y exclusión para filtrar artículos con mayor relevancia.
d) *Búsqueda de palabras claves.* Los revisores leen los resúmenes y buscan palabras clave y conceptos que reflejen la contribución del artículo [35].

e) *Extracción de datos y mapeo de estudios.* Una vez establecido el esquema de clasificación, se ordenan los artículos pertinentes en el esquema, es decir, se realiza la extracción de datos propiamente dicha [35].

2.6 Técnicas de procesamiento y análisis de datos

Para el análisis y procesamiento de los datos se aplicó la técnica estadística descriptiva, cuyo objetivo es el de resumir los datos del estudio mediante la recolección, análisis y caracterización de la información. Se utilizaron tablas como mecanismo para organizar la información de manera codificada y gráficos estadístico para resumirla visualmente de tal manera que facilitó el análisis. También se aplicó la técnica de análisis de patentes y literatura científica, la cual hace uso de los metadatos de investigaciones científicas y patentes facilitando discernir información sobre tendencias y relaciones entre estudios, autores o propiedad intelectual.

2.7 Normas éticas

El estudio se desarrolló siguiendo normativas estipuladas en el reglamento de grado de la Pontificia Universidad Católica del Ecuador, sede Esmeraldas (PUCESE), con la finalidad de que la investigación se enmarque en los lineamientos de la institución. También se respetó la propiedad intelectual de todas las ideas, trabajos, resultados e innovaciones que se han encontrado en la literatura, se ha citado al autor correspondiente por cada idea o afirmación utilizada en el estudio. Por otro lado, los resultados de este estudio serán inéditos ya que no han sido abordados en la literatura.

CAPITULO 3: RESULTADOS

3.1 Mapeo sistemático de inyecciones de fallos

En este punto se detalla el protocolo y resultados obtenidos del proceso de mapeo sistemático.

3.1.1 Protocolo del mapeo

3.1.1.1 Preguntas de investigación

Las siguientes preguntas de investigación basadas en el objetivo principal del estudio guiaron el proceso de mapeo sistemático.

Q1: ¿Qué técnicas, métodos o herramientas son utilizadas para llevar a cabo procesos de inyecciones de fallos?

Q2: ¿Cómo se pueden clasificar las técnicas de inyección de fallos?

3.1.1.2 Proceso de búsqueda

Para llevar a cabo la búsqueda se definió la formula en base a los términos de mayor relevancia del tema en cuestión, la Tabla *3* representa las palabras claves:

Tabla 3. Términos de búsqueda.

TERMINO 1	TERMINO 2	TERMINO 3
Técnicas	Inyección de fallos	Sistemas distribuidos
Techniques	Fault injection	System distributed
Methods	Failure injection	

Posteriormente se utilizaron servicios como SCOPUS, IEEEXPLORE, ACM y EBSCOHOST para recabar la información con la cadena de búsqueda. También es importante mencionar que se agregaron ciertos artículos de interés de forma manual de los servicios de Google académico, en la Tabla *4* se presentan las fórmulas utilizadas finales con exclusiones e inclusiones:

Tabla 4. Cadenas de búsquedas.

SCOPUS
TITLE (techniques) AND TITLE-ABS-KEY ("fault injection") OR TITLE-ABS-KEY ("failure injection") AND TITLE-ABS-KEY (systems)) AND (

LIMIT-TO (PUBYEAR , 2021) OR LIMIT-TO (PUBYEAR , 2020) OR LIMIT-TO (PUBYEAR , 2019) OR LIMIT-TO (PUBYEAR , 2018) OR LIMIT-TO (PUBYEAR , 2017) OR LIMIT-TO (PUBYEAR , 2016) OR LIMIT-TO (PUBYEAR , 2015))
IEEEXPLORE
"All Metadata":techniques) AND ("Document Title":"fault injection") OR ("Document Title":"failure injection")
ACM
[All: techniques] AND [[Publication Title: "fault injection"] OR [Publication Title: "failure injection"]] AND [All: systems] AND [Publication Date: (01/01/2015 TO 06/30/2021)]
EBSCOHOST
TI techniques AND TX "fault injection" OR TX "failure injection" AND TX "systems distributed"

3.1.1.3 Selección de estudios

En este punto se detallaron los criterios que permitieron excluir e incluir la información permitiendo filtrar artículos con mayor impacto o relevancia para el tema en cuestión.

Se consideró información relevante y se incluyeron aquellos que:

- Muestren evidencia de una aplicación de inyección de fallos. Esto descarta propuestas que no incluyan los resultados obtenidos luego del proceso de inyección de fallos.
- Información referente a procesos de inyección de fallos realizadas en los últimos 6 años (2015-2021).
- El enfoque del artículo esté dentro del área de la informática
- Se utilizará información de preferencia en inglés, y en caso de ser sustancial español.
- Permitido el acceso al texto en su totalidad a través de los motores de búsqueda antes mencionados.

Igualmente, los estudios que presentaron información importante para el estudio y no fueron detectados por las fuentes de búsqueda utilizadas, se agregaron a la bibliografía manualmente por el investigador.

Por consiguiente, no se consideraron relevantes y fueron excluidos aquellos resultados de búsquedas que no se apegaban a los criterios de inclusión antes mencionados.

3.1.1.4 Proceso de clasificación

A continuación, se llevó a cabo la metodología de clasificación denominada Keywording [36], la cual hace posible aminorar el tiempo para definir un esquema de clasificación y garantizar que ésta contemple la información procesada en el mapeo. Así mismo, se procedió con las lecturas de los resúmenes para extracción de términos claves y definiciones que evidencien la contribución al estudio. Al realizar la búsqueda ciertas definiciones fueron combinadas a partir de los artículos permitiendo un mejor entendimiento sobre el aporte de los resultados de la búsqueda. Esta metodología fue empleada para responder la interrogante Q2 (clasificación de las técnicas de inyección) basándose en el siguiente esquema de clasificación:

- Hardware (HWIFI)
- Software (SWIFI)
- Simulación (SFI)
- Emulación (EFI)
- Híbrida

3.1.2 Resultado del proceso de mapeo sistemático

La búsqueda se ejecutó en septiembre del 2020 en varios servicios de bases de datos documentales y bibliotecas digitales antes mencionadas, cuya resultante preliminar fueron 855 entradas posibles. Luego de ejecutar un primer proceso de filtrado basado en los criterios de inclusión sobre título, resumen, palabras claves y año, se obtuvieron 301 artículos para un posterior análisis con mayor detalle. Ahora bien, como resultante de un segundo proceso de filtrado sobre el texto completo de los estudios, se encontraron 84 artículos que describían inyecciones de fallos. Por otro lado, se agregaron manualmente 5 estudios del servicio de Google académico seleccionados por considerarse información relevante para el estudio, por consiguiente, en la Tabla *5* se presentan a detalle de los datos antes mencionados:

Tabla 5. Detalle de artículos por fuente

FUENTE	#ARTÍCULOS	FILTRO 1	FILTRO 2	REVISIÓN	SELECCIÓNADOS
SCOPUS	253	96	29	-	17
IEEEXPLORE	286	123	26	-	19
ACM	245	64	20	-	17
EBSCOHOST	56	18	12	-	5
GOOGLE ACADÉMICO	6	-	-	6	2
Total	**846**	**301**	**87**	**6**	**60**

Seguidamente, se analizan los resultados de la clasificación presentada para contestar la interrogante Q2. Se evidenció que la mayoría de las inyecciones de fallos que se llevaron a cabo en los años planteados son las basadas en software (31%) y por simulación (38%), por otra parte, pero en menor proporción las demás técnicas de inyecciones representadas en la Tabla *6*.

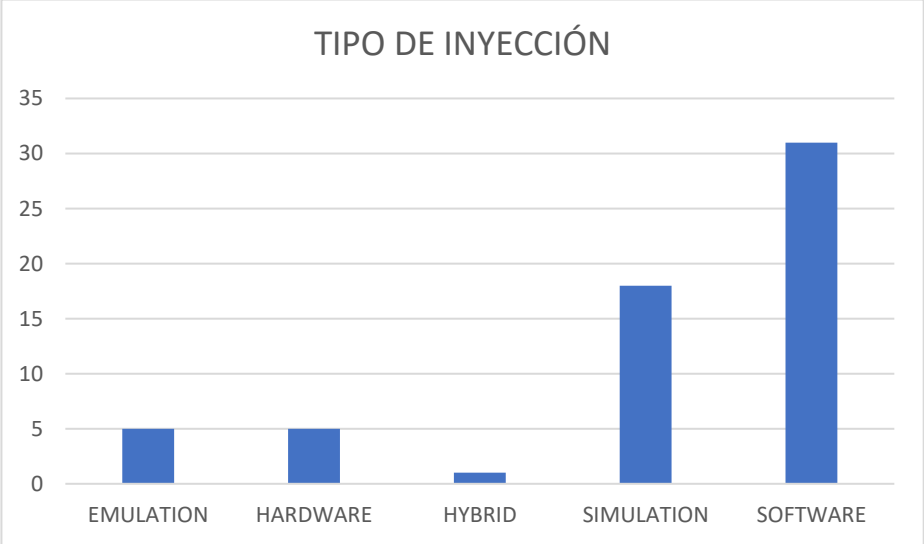

Figura 5. Cuenta de tipos de inyecciones

Agregando a lo anterior, dada la generalización de los sistemas distribuidos y todo lo que estos implican, se realizó una clasificación en varios campos en donde se aplicaron procesos de inyecciones de fallos. Se puede apreciar que en mayor proporción se dan en donde predomina el hardware como tal, siendo el caso de los circuitos y los sistemas embebidos (18% y 17%) respectivamente.

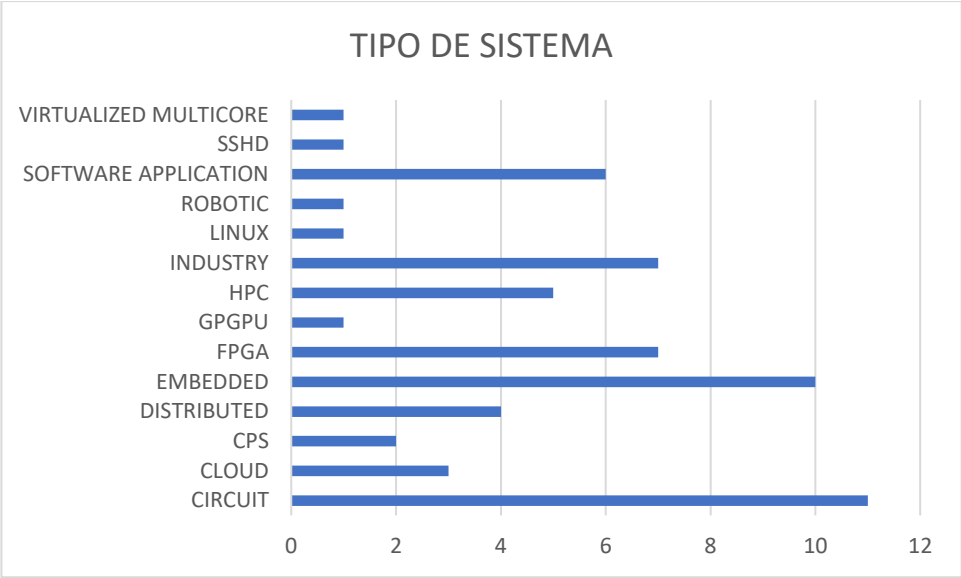

Figura 6. Cuenta de tipos de sistemas.

El detalle de todos los resultados del mapeo se puede encontrar en el anexo 1.

3.2 Comparación de las técnicas de inyección de fallos

La caracterización de una técnica de inyección de fallos es va a depender de un grupo de criterios de orden científico (objetivos de la experiencia), técnico (fase del periodo de vida, tecnología usada) y económico (coste). Cada uno de estos criterios, fueron representados en Tabla *6*, y ponderados por medidas (alto, medio, baja), por cada técnica sujeta a comparación. Para un mejor entendimiento del por qué un parámetro en una técnica determinada es (alto, medio, baja) se detalla a continuación:

3.2.1 Genericidad

En las inyecciones de fallos basadas en simulación existe una tendencia en la determinación de formalismos que comprendan una gran variedad de modelos, lo que acrecienta notablemente la genericidad de los procesos de inyección de fallos.

La implementación de técnicas basadas en hardware se limita a casos en los que se cuenta con un modelo físico (prototipo), y su puesta en marcha está condicionada cuando el desarrollo está en las fases finales. No obstante, permiten la comprobación del sistema real en su totalidad.

Las técnicas (SWIFI) presentan una limitación parecida, puesto que su aplicación se basa en sistemas que corren programas. Estas técnicas solo se pueden llevar a cabo en instancias finales del diseño, cuando existe interoperabilidad entre hardware y software; consiste, de hecho, de correr el software en su ambiente operacional. Entre los mecanismos de inyección y su entorno existen dependencias, las cuales conllevan un alto grado de especificaciones de la herramienta utilizada con relación de un determinado sistema. Esto implica como consecuencia evidente, el grupo de herramientas expuestas en la bibliografía las cuales se alinean a un sistema particular.

3.2.2 Accesibilidad y controlabilidad

En teoría, la inyección basada en simulación tiene una accesibilidad total, esto debido a que el modelo utilizado es una estructura de datos. Siempre un proceso de simulación tendrá un nivel de accesibilidad y controlabilidad alto, esto porque las pruebas que se realicen o los procesos de inyecciones que se ejecuten serán sobre un modelo ya existente, que está en funcionamiento.

La accesibilidad en las técnicas basadas en hardware va a depender del lugar en el cual se inyecta y los métodos utilizados, por ejemplo: el esqueleto o la forma en como está estructurado un circuito integrado encapsulado tiende a ser inabordable a las formas utilizadas para llevar a cabo inyecciones a nivel de pin (mediante sondas o soportes especiales), no obstante, es totalmente abordable a procesos de inyección por radiación.

En otro sentido, en la inyección basada en software se pretende preparar al sistema en una situación preciso o determinada, que se asemeje a la ocurrencia no controlada de un fallo. Planteando un ejemplo de un procesador, las limitaciones de accesibilidad serían; por el impedimento de determinar el contenido de algunos registros internos por medio del conjunto de instrucciones, de colocar un par de señales tales como READ y WRITE en un estado ilógico (READ y WRITE sincrónicamente activas) o de efectuar operaciones sobre el bus descoordinadas con el reloj.

3.2.3 Neutralidad

Por lo general la neutralidad de las técnicas de inyección es escasa, pero tienden a diferir levemente entre ellas. Esto indica, la necesidad de las técnicas a un punto medianamente

directo para los procesos de inyección. En la inyección basada en hardware, esto se hace evidente en la inyección a nivel de pin por inserción (se debe extraer el circuito de la placa, lo cual sería un inconveniente si está soldado) y por otro lado (el circuito tiene que ser situado en un lugar adecuado) en el caso de bombardeo con iones pesados.

En general en los procesos de inyección basadas en simulación, el modelo no tiende a estar sujeto a la aparición de mutaciones parásitas no detectadas, tampoco a introducciones de fallos duros.

Los cambios para llevar a cabo procesos de inyección basados en software suceden en los programas de la aplicación o en las herramientas de explotación. Estos cambios tienden a ser la causa de mutaciones parásitas. La implantación de determinadas instrucciones de la inyección cambia el tiempo de ejecución del código. Pero, no hacen posible la inserción de fallos invariables como las técnicas basadas en simulación.

3.2.4 Automatización

Es mucho más fácil manipular un modelo de simulación en comparación con uno físico, debido a que la estipulación de los parámetros de inyección, la ubicación de los inyectores y el trato de resultados solamente requiere tratamiento de datos informáticos.

Los procesos basados en HWFI presentan una limitante a la hora de automatizar determinadas técnicas, como es el caso de inyecciones a nivel de pin, en la cual se requiere mover la sonda de inyección. Al contrario, el proceso de inyección por bombardeo de iones pesados es muy automatizable. Contrastando, otras operaciones como (inyección, inicialización del sistema, medidas, entre otras.) logran ser ejecutadas automáticamente por el directorio de pruebas.

Finalmente, en las SWIFI se utilizan tipos de datos parecidos a los utilizados por técnicas basadas en simulación, por lo que el nivel de automatización de ambas es similar.

3.2.5 Precisión

Es limitada la precisión que se pueden alcanzar mediante técnicas de inyección basadas en simulación, en primera instancia, por la función de modelado de fallos y en segunda instancia, debido a la expansión de los dispositivos de inyección que presenta. De manera

general, se puede afirmar que la correspondencia entre los modelos de fallos físicos y los fallos inyectados se establece preferentemente en los niveles de modelado más bajos.

La inyección implementada mediante hardware favorece naturalmente la equivalencia, directa o indirecta, entre los fallos inyectados y los reales encontrados durante la vida operacional del sistema. Esta equivalencia es prácticamente absoluta en el caso de técnicas de inyección por radiación; en efecto, los fallos inyectados son de naturaleza similar a los engendrados por radiaciones iónicas encontradas en la atmósfera (a nivel de componentes de procesadores situados en satélites, por ejemplo).

En la inyección HWIFI a nivel de pin, la equivalencia es indirecta. Los fallos de rotura de pista o cortocircuito a la alimentación tienen tendencia a desaparecer gracias a la mejora del proceso de fabricación. Los fallos inyectados deben representar entonces los fallos internos a los circuitos. Si esta representatividad es corroborada en cierta medida por la experiencia, es sin embargo interesante constatar que los modelos de fallos de stuck-at inyectados en la periferia de los circuitos son los mismos en el caso de circuitos MSI (puertas lógicas) que en circuitos de más alta escala de integración; está claro que la precisión no puede ser la misma en ambos casos. De hecho, la hipótesis que sostiene la técnica de inyección a nivel de pin de que la mayoría de los fallos internos se producen a nivel de circuitos de interfaz, y se manifiestan como fallos stuck-at a nivel de pin, debe necesariamente ser revisada para tener en cuenta el aumento exponencial de la densidad de integración.

Desde el punto de vista de la precisión, las técnicas de inyección SWIFI presentan cierta similitud con las HWIFI. En efecto, la búsqueda de una equivalencia entre los efectos de los fallos inyectados y los de los fallos físicos se basa en los esfuerzos de modelado de estos efectos a nivel del software, pero también en las limitaciones propias de la aproximación, como la controlabilidad.

3.2.6 Coste

Para poder analizar qué tan costoso es llevar a cabo un proceso de inyección de fallos de una técnica en particular, se debe tomar en consideración el ámbito económico y el temporal. Las técnicas de inyección basadas en modelos analíticos o en simulación sólo utilizan los programas de tratamiento de la información contenida en los modelos sobre los que se aplican. Por esta razón, el coste económico de la herramienta comprende

esencialmente el coste del entorno de simulación. En cuanto al coste temporal asociado al desarrollo de la herramienta, éste es relativamente reducido, ya que en principio sólo hay que desarrollar una aplicación.

El coste para llevar a cabo un proceso de inyección basadas en hardware va a depender del método utilizado. Las herramientas basadas en la inyección de iones pesados son posiblemente las más costosas. En la inyección por exposición es necesaria una fuente radiactiva (difícil de adquirir, tanto por su coste económico como por los especiales pasos que hay que seguir), y en la inyección mediante el bombardeo con un acelerador (siendo preciso alquilar la instalación, con el consiguiente coste económico, además de las restricciones temporales). En cuanto a las herramientas de inyección a nivel de pin, algunas de ellas están disponibles comercialmente, lo que en caso de resultar satisfactorias reducen a cero el coste de su realización. El resto de las técnicas se encuentran entre ambos extremos.

Por su naturaleza puramente lógica, el coste de una herramienta SWIFI es escaso: es el coste del soporte. Del mismo modo, los costes de preparación y de inyección son reducidos, sabiendo que los mecanismos de inyección son, en general, específicos a un procesador y un sistema de explotación dados. Por último, ya que una campaña de inyección da lugar de forma casi sistemática a un conjunto de datos (almacenados durante las experiencias), el coste ligado a su tratamiento es generalmente el mismo para todas las técnicas de inyección, sean físicas o simuladas.

Tabla 6. Comparación de las técnicas de inyección.

	TÉCNICA		
	HARDWARE	**SOFTWARE**	**SIMULACIÓN**
GENERICIDAD	Media	Baja	Alta
ACCESIBILIDAD Y CONTROLABILIDAD	Media	Media	Alta
NEUTRALIDAD	Baja	Media	Alta
AUTOMATIZACIÓN	Media	Alta	Alta
PRECISIÓN	Alta	Media	Baja
COSTE	Alta	Bajo	Media

CAPITULO 4: DISCUSIÓN

En la actualidad, la necesidad de contar con sistemas altamente fiables implica que los desarrolladores adopten diferentes procesos que les permitan abarcar o en tal caso prepararse a fallos no contemplados, permitiéndoles una rápida recuperación a estos eventos y así poder aminorar el impacto de las consecuencias que implican. Por tal motivo adoptan técnicas de inyección de fallos en sus procesos de desarrollo. En la literatura se han identificado varias técnicas, métodos y tecnologías con las que se pueden llevar a cabo las inyecciones de fallos, satisfaciendo el desarrollo del primer objetivo del estudio.

Según la investigación [13], en la cual los autores presentan un enfoque basado en la técnica SWIFI, aplicando el aprendizaje mecánico no supervisado en los rastros de ejecución del sistema inyectado y apoyándose de la plataforma OpenStack para las experimentaciones, llegaron a la conclusión que los costos computacionales son bajos asemejándose a los resultados expuestos en la tabla 6 del estudio, en donde los mecanismos de inyección son, en general, específicos a un procesador y un sistema de explotación dados.

Otro aspecto según [20], en el cual detallan un proceso HWIFI para medir la robustez de un sistema inyectando fallos tanto transitorios y permanentes permitiéndoles medir la fiabilidad, hace posible connotar la complejidad que implica llevar a cabo está técnica la cual va a depender del lugar en el que se inyectan los fallos y los métodos utilizados, por ende, se puede concluir que tienen un nivel de accesibilidad y controlabilidad medio.

Considerar también la herramienta presentada por [14], con la cual es posible llevar a cabo procesos de inyecciones SFI. Evaluaron las fallas de memoria del sistema operativo basado en Unix sometiéndolo a presión por programas que utilizaban varios elementos del núcleo Linux, contrastando con los resultados de este estudio se entiende que los niveles de accesibilidad y controlabilidad siempre será alta, ya que las pruebas que se llevan a cabo son sobre modelos ya existentes y que están en funcionamiento. También hay que considerar que por lógica la neutralidad de esta técnica por lo general será escasa ya que la aparición de fallas siempre es controlada. En forma similar el estudio [3], en el cual aconsejan implementar inyecciones en las fases de iniciales de desarrollos de los sistemas. Presentaron un marco que simula modelos de fallas y que admite componentes OTS. Esto permite mayor facilidad de automatización de un modelo de simulación en comparación con uno físico. En cuanto a los costos estos procesos tienden a ser

considerablemente bajos debido a que solo utilizan los programas del tratamiento de la información. Hay que considerar que esta técnica a pesar de contar con altos niveles de accesibilidad, genericidad, controlabilidad y sobre todo su bajo costo de implementación, se ven contrastadas por el bajo nivel de precisión que tienen.

CAPITULO 5: PROPUESTA

5.1 Título

Taxonomía de técnicas de inyecciones de fallos.

5.2 Representación

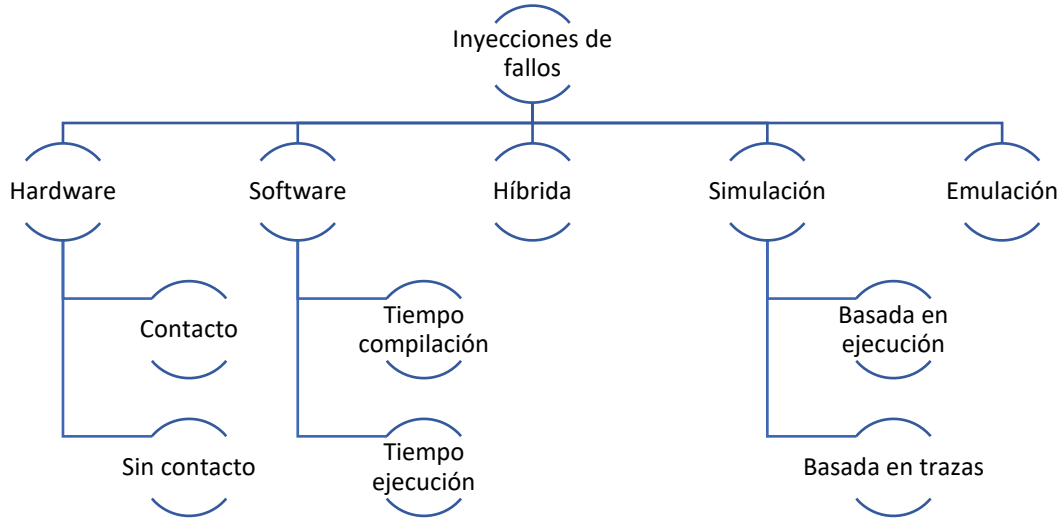

Figura 7. Representación gráfica de la taxonomía.

5.3 Descripción

En esta taxonomía se presentan las formas en cómo es posible llevar a cabo un proceso de inyección de fallos. En la literatura es muy común encontrar aplicaciones de las técnicas basadas en hardware, simulación y software, siendo esta última la más utilizada debido a su bajo costo de implementación. Pero, por lo general son aplicadas de manera independiente. Sin embargo, durante el estudio se identificaron técnicas basadas en emulación la cual se presenta como una mejora a la inyección basada en simulación, y la

técnica híbrida la cual es la aplicación combinada de varias de las técnicas antes descritas, punto que sirvió de motivación para la presentación de esta propuesta.

Se agruparon las cinco técnicas de inyecciones de fallos, luego de ser identificadas, estudiadas y comparadas como un solo conjunto, lo cual facilitará a los desarrolladores identificar que técnica aplicar en sus sistemas, mediante la comparación antes descrita sabrán que tan beneficioso es adoptar una técnica u otra, más aún, que herramientas se han venido utilizando y con cuales se han obtenido mejores resultados. También, se debe tomar en cuenta que los sistemas distribuidos son por lo general grandes y complejos y lo que se prioriza es que funcionen la mayor parte del tiempo sin interrupción alguna, en otras palabras, alta fiabilidad.

Esta propuesta servirá como preámbulo para estudios mucho más profundos en una técnica determinada, o para estudios comparativos que abarquen las cinco técnicas antes descritas. En síntesis, la propuesta está direccionada a conseguir un mejor entendimiento de lo que implica un proceso de inyección de fallos y sobre todo al desarrollo de sistemas cuya necesidad es tener un mayor grado de fiabilidad.

Considerar también, la propuesta del estudio se planteó en base a información referente a los últimos seis años, por lo que se podrá decir que en la investigación se describen tendencias, mejoras en cada uno de los procesos de las técnicas y últimas herramientas utilizadas, información que se podrá evidenciar en el mapeo sistemático y descrita debidamente en los resultados del estudio.

CAPITULO 6: CONCLUSIONES

Esta investigación documental presenta el análisis de estudios orientados a procesos de técnicas de inyecciones de fallos, en una variedad de sistemas los cuales están comprendidos dentro de la temática sistemas distribuidos, y se obtuvieron las siguientes conclusiones:

- Se identificaron cinco técnicas basadas en (hardware, software, simulación, emulación e híbrida) mediante la aplicación de un mapeo sistemático, para llevar a cabo procesos de inyecciones de fallos, cuyas aplicaciones van a depender del sistema objeto. También, se determinaron varias herramientas propuestas por autores que apoyadas de diferentes tecnologías permiten crear sistemas con un mayor grado de fiabilidad.
- Se compararon las técnicas antes mencionadas en base a los parámetros de genericidad, accesibilidad y controlabilidad, neutralidad, automatización, precisión y coste, lo cual permitirá a desarrolladores esclarecer que tan conveniente es aplicar una u otra técnica. Por consiguiente, se presentó una tabla para una mejor comprensión de estos resultados estipulados según ponderaciones alta, media, bajo, referenciados a los parámetros.
- Finalmente, se presentó como propuesta la taxonomía de clasificación de las técnicas de inyección de fallos, ya que no han sido abordadas en la literatura como un conjunto de técnicas, si no que por lo general son procesos que se aplican de manera independiente lo cual no facilitaba una comparación entre ellas, esto hasta antes de llevar a cabo este estudio.

CAPITULO 7: RECOMENDACIONES

Considerar esta investigación para ampliar el entendimiento sobre los procesos de inyecciones de fallos, facilitando decidir la que mejor convenga para el sistema al que se desee aplicar, también puede servir como preámbulo para posteriores comparativas entre técnicas basadas en parámetros parecidos a los presentados.

Se recomienda aplicar inyecciones de fallos basadas en software en casos en los que los recursos son limitados, debido a los bajos costos de implementación y al sinnúmero de herramientas con las que se pueden implementar, siendo la técnica mayormente utilizada, dato evidenciado en el proceso del mapeo sistemático del estudio.

Se sugiere considerar procesos de inyecciones de fallos basados en software en sistemas críticos, los cuales una interrupción en el normal funcionamiento puede resultar en pausa de servicios básicos, pérdidas económicas o en el peor de los casos pérdidas humanas. Esta técnica suele tener un coste elevado tanto temporal como económico ya que se aplica sobre el sistema real, y la complejidad del proceso conlleva un mayor esfuerzo, todo esto dando como resultante sistemas con alto grado de fiabilidad.

Las técnicas de inyección basadas en simulación se aconseja implementarlas en sistemas en donde el tiempo para el proceso de desarrollo es reducido y los recursos son limitados, esto permitirá abarcar un conjunto de posibles falencias en el sistema, pero no en su totalidad.

Referencias

[1] G. Juan and A. Mart, “Tecnicas Para La Inyeccion De Defectos De Hardware Y Software Bajo El Sistema Operativo GNU / Linux Informe de proyecto de grado,” 2010.

[2] D. Juan Carlos Baraza Calvo, “Contribución a la validación de sistemas complejos tolerantes a fallos en la fase de diseño. Nuevos modelos de fallos y técnicas de inyección de fallos,” *UNIVERSIDAD POLITÉCNICA DE VALENCIA DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES*, 2003.

[3] A. Höller, G. Macher, T. Rauter, J. Iber, and C. Kreiner, “A Virtual Fault Injection Framework for Reliability-Aware Software Development,” *Proceedings - 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2015*, pp. 69–74, 2015, doi: 10.1109/DSN-W.2015.16.

[4] J. C. Becker and G. Flick, “Practical approach to Failure Mode, Effects and Criticality Analysis (FMECA) for computing systems,” *Proceedings of the High-Assurance Systems Engineering Workshop*, pp. 228–236, 1997, doi: 10.1002/j.2334-5837.1996.tb01996.x.

[5] A. Çelik *et al.*, *DISTRIBUTED SYSTEMS Concepts and Design*, vol. 1, no. 1. 2018.

[6] A. Bolfing and A. Bolfing, *Distributed Systems*, vol. 01. 2020. doi: 10.1093/oso/9780198862840.003.0005.

[7] A. S. Tanenbaum, *Sistemas Distribuidos / Principios y Paradigmas / 2da ed*. 2008.

[8] F. D. Muñoz Escoí *et al.*, *Concurrencia y sistemas distribuidos*. 2012.

[9] A. S. Tanenbaum, *Sistemas Distribuidos / Principios y Paradigmas / 2da ed*. 2008.

[10] F. de A. López Fuentes, *Sistemas Distribuidos*, vol. 1, no. 1. 2015.

[11] G. Coulouris, J. Dollimore, and T. Kindberg, “Sistemas Distribuidos: Conceptos y Diseño,” p. 726, 2007.

[12] W. A. Lucero, C. Salgado, A. Sánchez, and M. Peralta, “Fiabilidad en la Calidad del Software : Modelos , Métodos y Estrategias,” 2020.

[13] D. Cotroneo, L. De Simone, P. Liguori, and R. Natella, “Fault Injection Analytics: A Novel Approach to Discover Failure Modes in Cloud-Computing Systems,” *IEEE Transactions on Dependable and Secure Computing*, vol. 5971, no. c, pp. 1–16, 2020, doi: 10.1109/TDSC.2020.3025289.

[14] A. D. Velasco, B. Montrucchio, and M. Rebaudengo, “KITO tool: A fault injection environment in Linux kernel data structures,” *Microelectronics Reliability*, vol. 60, pp. 153–162, 2016, doi: 10.1016/j.microrel.2016.02.011.

[15] M. Kooli and G. Di Natale, “A survey on simulation-based fault injection tools for complex systems,” *Proceedings - 2014 9th IEEE International Conference on Design and Technology of Integrated Systems in Nanoscale Era, DTIS 2014*, 2014, doi: 10.1109/DTIS.2014.6850649.

[16] M. Cinque and A. Pecchia, “On the injection of hardware faults in virtualized multicore systems,” *Journal of Parallel and Distributed Computing*, vol. 106, pp. 50–61, 2017, doi: 10.1016/j.jpdc.2017.03.004.

[17] M. Kooli, F. Kaddachi, G. Di Natale, A. Bosio, P. Benoit, and L. Torres, “Computing reliability: On the differences between software testing and software fault injection techniques,” *Microprocessors and Microsystems*, vol. 50, pp. 102–112, 2017, doi: 10.1016/j.micpro.2017.02.007.

[18] D. Ferraretto and G. Pravadelli, “Efficient fault injection in QEMU,” *2015 16th Latin-American Test Symposium, LATS 2015*, 2015, doi: 10.1109/LATW.2015.7102401.

[19] T. Given-Wilson, N. Jafri, and A. Legay, “Combined software and hardware fault injection vulnerability detection,” *Innovations in Systems and Software Engineering*, vol. 16, no. 2, pp. 101–120, 2020, doi: 10.1007/s11334-020-00364-5.

[20] R. M. Tawfeek, M. G. Egila, Y. Alkabani, and I. M. Hafez, “Fault injection for FPGA applications in the space,” *Proceedings of ICCES 2017 12th International Conference on Computer Engineering and Systems*, vol. 2018-Janua, pp. 390–395, 2018, doi: 10.1109/ICCES.2017.8275338.

[21] J. Wei, A. Thomas, G. Li, and K. Pattabiraman, “Quantifying the accuracy of high-level fault injection techniques for hardware faults,” *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 375–382, 2014, doi: 10.1109/DSN.2014.2.

[22] M. Ebrahimi, M. Rashvand, F. Kaddachi, M. B. Tahoori, and G. Di Natale, “Revisiting software-based soft error mitigation techniques via accurate error generation and propagation models,” *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design, IOLTS 2016*, pp. 66–71, 2016, doi: 10.1109/IOLTS.2016.7604674.

[23] M. Liu, Z. Zeng, and J. Cai, “Embedded Software Based on JTAG Interface,” 2016, doi: 10.1109/ICRMS.2016.8050155.

[24] R. Velazco, D. Mcmorrow, and J. Estela, *pace*. 2019.

[25] R. Piscitelli, S. Bhasin, and F. Regazzoni, “Fault attacks, injection techniques and tools for simulation,” *Hardware Security and Trust: Design and Deployment of Integrated Circuits in a Threatened Environment*, pp. 27–47, 2017, doi: 10.1007/978-3-319-44318-8_2.

[26] R. Velazco, D. Mcmorrow, and J. Estela, *pace*. 2019.

[27] D. Juan Carlos Baraza Calvo, “Contribución a la validación de sistemas complejos tolerantes a fallos en la fase de diseño. Nuevos modelos de fallos y técnicas de inyección de fallos,” *UNIVERSIDAD POLITÉCNICA DE VALENCIA DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES*, 2003.

[28] D. de Informática De Sistemas Y Computadores Validación Por Inyección De Fallos En Vhdl De La Arquitectura Tta and P. D. por Joaquín Gracia Morán Dirigida por D Daniel Antonio Gil Tomás D Pedro Joaquín Gil Vicente Valencia, “VALIDACIÓN POR INYECCIÓN DE FALLOS EN VHDL DE LA ARQUITECTURA TTA,” 2004.

[29] M. Portela, G. Directora, D. Celia, and L. Ongil, “Técnicas de inyección de fallos basadas en FPGAs para la evaluación de la tolerancia a fallos de tipo SEU en circuitos digitales,” 2007.

[30] C. Fibich, S. Tauner, P. Rössler, M. Horauer, M. Matschnig, and H. Taucher, “FIJI: Fault InJection Instrumenter,” *Eurasip Journal on Embedded Systems*, vol. 2019, no. 1, 2019, doi: 10.1186/s13639-019-0088-7.

[31] Propiedad Intelectual Registro Oficial No 320, “Registro Oficial No 320 Ley de Propiedad Intelectual,” *2015*, no. 320, p. 92, 2015.

[32] Consejo de Educación Superior, “Ley Organica De Educacion Superior, LOES,” *2018*, pp. 1–58, 2018.

[33] Constitución, “Constitución del Ecuador,” *Registro Oficial*, no. 20 de Octubre, p. 173, 2008.

[34] D. Causas, “Definición de las variables , enfoque y tipo de investigación,” *Universidad Nacional Abierta y a Distancia (UNAD)*, pp. 1–11, 2005.

[35] M. Learning and R. Cookbook, *Metodología de la investigación*. 2014.

[36] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” *12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008*, no. February 2015, 2008, doi: 10.14236/ewic/ease2008.8.

ANEXOS

Plantilla de los resultados del mapeo sistemático: https://puceseedu-my.sharepoint.com/:x:/g/personal/luis_torres_pucese_edu_ec/ESJvG5myjG9FnoxTscUh4G4BuHHAEFgPdu-bMcGgSnJuAg?e=BmYlak