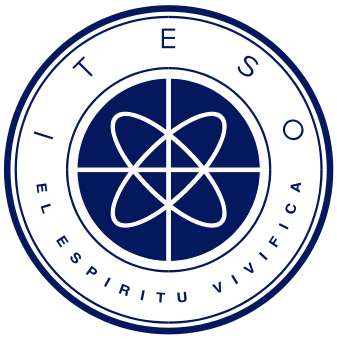
Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Informática Aplicada



STARLINGX

PROYECTO DE VALIDACIÓN PARA SISTEMAS *EDGE CLOUD* EN TECNOLOGÍAS *OPEN SOURCE* DE INTEL

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRA EN INFORMÁTICA APLICADA

Presenta: Juan Pablo Gomez Preciado

Asesor: Parres Peredo Alvaro Ivan

TLAQUEPAQUE, JALISCO. 14 JUNIO DE 2025

HISTORIAL DE REVISIONES

Revisión #	Fecha de Revisión	Descripción del Cambio	Autor
1	13/04/20	Inicio de Documentación	JP
2	14/05/20	Modificación de Documentación	JP
3	10/10/20	Inicio de Marco de Referencia	JP
4	06/12/22	Integración de Mejoras para IDI4	JP
5	30/06/25	Ultima Revision	Tutor Alvaro
6	14/07/25	Version Final	JP

TABLA DE CONTENIDO

HISTORIAL DE REVISIONES 1
RESUMEN 3
1. INTRODUCCIÓN 8
2. MARCO TEÓRICO 9
3. DESCRIPCIÓN DEL PROYECTO 23
2.1 ANTECEDENTES DEL PROYECTO 23
2.2 OBJETIVO DEL PROYECTO 27
2.3 DESCRIPCIÓN METODOLÓGICA 28
2.4 PLANEACIÓN CRONOLÓGICA DEL PROYECTO 34
2.5 DESCRIPCIÓN DE LAS ACTIVIDADES 35
2.6 RESUMEN DE LA DOCUMENTACIÓN E INFORMACIÓN RECABADA 43
2.7 RESULTADOS OBTENIDOS EN EL PROYECTO REPORTADO 44
4. CONCLUSIONES 46
4.1 LECCIONES APRENDIDAS 46
4.2 PROPUESTA DE MEJORA 47
4.3 CONCLUSIONES 52
5. BIBLIOGRAFÍA 54
Bibliography 54
6. ANEXOS 56

RESUMEN

En la actualidad, nos encontramos en la era de los servicios, donde plataformas digitales, pequeñas y grandes empresas, han implementado cambios importantes en sus organizaciones para posicionarse no solo como proveedores de tecnología, comida o educación, sino también como proveedores de servicios.

La nube actualmente es uno de los servicios de mayor uso y con mayores implementaciones tecnológicas. En 2018 Intel tomo la decisión de colaborar en un proyecto *open source* para sistemas de nube tipo *edge*. Dentro del mundo del *open source*, la colaboración es lo más importante e Intel siempre se ha preocupado en colaborar en proyectos alrededor del mundo como: Linux Kernel, Clear Linux, Deep Learning Software Stacks y StarlingX.

StarlingX antes de ser llamado así, fue un proyecto para tecnologías cloud Edge iniciado por WindRiver, sin embargo, la apuesta de Intel fue colaborar en el mismo proyecto para tener presencia en el sector cloud de open source. Intel se integró a StarlingX con desarrollo, integración y validación. En este trabajo solo hablare del trabajo que se desarrolló en la validación del proyecto StarlingX.

StarlingX es una infraestructura completa de software para Edge Cloud usada por las aplicaciones más demandantes en la industria IOT, tales como: telecomunicaciones, video y casos de ultra baja latencia. Intel aposto por sistemas Cloud tipo el tipo Edge es de baja latencia, a raíz de que Edge trabaja más cerca de los dispositivos y al mismo tiempo con los recursos de cómputo y almacenamiento de datos, minimizando los tiempos de respuesta.

StarlingX inicialmente fue un proyecto que empezó con diferentes nombres tales como: Akraino y Madawaska, en colaboración con empresas como WindRiver y OpenStack, el proyecto se encontraba dividido en

diferentes países Canadá, China, y Estados Unidos, después de un tiempo también se unió Alemania.

Intel tomo la decisión de crear este proyecto de validación del código de StarlingX en la localidad de Intel de Guadalajara en Jalisco, México.

Este proyecto necesitaba una base sólida de validación para empezar a funcionar, por lo cual se empezó a trabajar con la integración de pruebas funcionales y de sistemas nuevas para poder validar todos los nuevos componentes de este proyecto.

A lo largo del 2019, se iniciaron procesos manuales para generar pruebas nuevas e integrar pruebas existentes de otros proyectos de nube. Con el tiempo se planeó integrar un framework open source para la automatización de las pruebas, ya que en un inicio no había automatización alguna y todos los procesos eran manuales.

Un *framework* de automatización de pruebas de acuerdo con SmartBear [1], son un set de guías y reglas usadas para crear y diseñar casos de pruebas. Estos *frameworks* de pruebas son parte esencial para el éxito de cualquier proceso automático de pruebas, dicha herramienta reduce los costos y tiempos de mantenimiento y esfuerzos para crear casos de prueba, agregando un retorno de inversión para los equipos de calidad buscando optimizar los procesos al ser más agiles.

Con el tiempo se decidió implementar una herramienta llamada Robot, la cual es un framework de validación open source para automatizar pruebas de diferentes funcionalidades, dicha herramienta está desarrollada en Python y actualmente se encuentra dentro de los 7 mejores frameworks open source.

La implementación del framework fue un éxito, se crearon suites para diferentes componentes, las cuales se podían ejecutar por prueba, por conjunto de pruebas, por suite o por componente.

Por otro lado, también se implementaron procesos de *Continous Integration/Continous Delivery* (CI/CD) para generar control en los cambios y hacer pruebas automáticas, después de haber sido integrados dichos cambios. Los procesos de CI/CD incorporan la automatización continua y por otro lado el control en todo el ciclo de vida del proyecto de validación.

A lo largo del tiempo, se iban generando mejoras con éxito a las suites de pruebas y en paralelo a los procesos de integración y desarrollo de pruebas, generando una suite robusta automatizada, con mayor contenido de casos de prueba para diferentes componentes.

Después haber implementado exitosamente dichas mejoras en el framework de pruebas automatizadas, procesos internos y de automatización de CI/CD, se detectaron nuevas necesidades:

- Aumentar la colaboración y visualización del proyecto StarlingX en la comunidad *Open Source*
- Soportar y validar nuevos cambios que provengan de nuevos desarrolladores de cualquier parte del mundo
- Generar ambientes virtuales para validar cambios en el código y al mismo tiempo reporte de resultados para el desarrollador
- Reducir la complejidad de generar ambientes de *cloud* para ser validados
- Reducción de tiempo para la curva de aprendizaje en entender el sistema de validación de esta nueva tecnología por cualquier desarrollador
- Escalabilidad de *Hardware*
- Integración de código de forma automática previamente validado por Intel y no por el desarrollador

Con el paso del tiempo se fueron cubriendo parte de dichas necesidades, sin embargo, por otro lado, existía aun la necesidad de encontrar la manera de maximizar los resultados fuera de Intel, específicamente en el sector de la comunidad open source.

Dada dichas necesidades se propuso implementar *Validation as a Service* (VaaS). La propuesta fue que dicha herramienta trabajaría como un servicio y se implementaría para el proyecto de StarlingX el cual maneja tecnologías en la nube para la comunidad *open source.*

Este servicio de validación pretendía exponer los servicios internos de validación para el proyecto de StarlingX de Intel Guadalajara a la comunidad open source para el mismo proyecto de StarlingX, generando integración automática y reporteo de errores a cualquier desarrollador del mundo.

Este servicio pretendía ayudar a colaborar, crear ambientes de prueba con infraestructura de Intel para validar el software de cloud de StarlingX en la organización de Intel en el área *Open Source Technology Center* (OTC) StarlingX provee hacia la comunidad basado en *OpenStack* y a la vez validar los cambios, modificaciones, e innovaciones de código que algún desarrollado fuera de Intel pueda generar.

De esta forma el desarrollador sabría si sus cambios funcionan correctamente con el código que acaba de integrar al proyecto StarlingX siendo sus cambios validados automáticamente por (VaaS).

Por otro lado, este servicio pretendía ayudar a simplificar el desarrollo de ambientes de prueba, reducir el tiempo y reducir el retrabajo tanto del desarrollador de la comunidad como para la validación interna del proyecto en Intel, dichos beneficios ayudarían por lo tanto a crear más contribuidores a nivel mundial, haciendo a StarlingX, aceptado e importante en la comunidad de cloud tipo Edge en open source y por ende en el mundo.

Sin embargo, el servicio de *Validation as a Service* no se llevó a cabo puesto que fue afectado por una reorganización interna, la cual detuvo todas las actividades nuevas por implementar, dejando el proyecto de validación solamente con validación interna, dentro de Intel. Dadas dichas circunstancias el proyecto duro en Guadalajara 2 años y después este fue transferido por completo a Alemania.

1. INTRODUCCIÓN

Dentro del mundo de IT existen un sinfín de herramientas que ayudan a desarrollar, probar e integrar proyectos. En este proyecto de validación de un sistema cloud edge computing como StarlingX se necesitaron también herramientas para desarrollar tales como frameworks de automatazion (Robot) y automatización (Jenkins), probar e integrar y de esta forma desarrollar la calidad confiable dentro de su código.

Edge Computing es una de las recientes tecnologías *cloud,* la cual ha tenido un importante impacto a nivel industrial, gracias a los bajos tiempos de respuesta de sus sistemas en la industria.

Edge computing como se menciona en [2], es una arquitectura cloud en donde la información y los datos del cliente son procesados lo más cerca posible al origen de dicha fuente de datos, eso agiliza el procesamiento y mejora la latencia en el manejo y proceso de datos e información.

En la actualidad nos podemos referir a *Edge computing* como la tecnología que ha ayudado a agilizar los procesamientos de datos en la nube.

Como caso de éxito podemos mencionar la universidad [3], en donde la Universidad imparte clases virtuales inmersivas en donde los alumnos pueden estudiar en *streaming* y desde sus propios dispositivos.

También APM Terminals [4] una de las grandes operadoras de diseños, operaciones de puertos, terminales marítimos y terrestres en todo el mundo se suma a la lista como muchos otros de casos de éxito del *Edge computing* gracias a su proyecto piloto. Telefónica desarrolla, en la terminal de contenedores APM en el puerto de Barcelona, una prueba piloto para manejar la seguridad en la terminal mediante una combinación de tecnologías como 5g y Edge Computing. Dichas tecnologías se utilizaron

para geolocalización y posicionamiento virtual de objetos fijos y al mismo tiempo geolocalización de elementos en movimiento. Este caso de éxito dio como resultado el control exacto entre los elementos fijos y en movimiento los cuales sin estas tecnologías tales como Edge computing no hubiera sido posible.

Dentro del proyecto de StarlingX surgió la necesidad de generar un proyecto de validación el cual empezó desde cero en el área de *open source* en Intel México.

Dicho proyecto de validación su objetivo principal era el de probar/validar el código de funciones y de sistemas para producir un código de alta calidad y confiabilidad para el proyecto StalirngX, si no existiera este proyecto de validación, simplemente el código fuente tendría muchos errores una vez que estuviera en producción con el cliente, lo cual ocasionaría: pérdida de tiempo en reparar errores y pérdida de confianza para Intel, lo que repercutiría en pérdida de ingresos económicos.

2. MARCO TEÓRICO

El sector de Open source a nivel mundial de acuerdo con Opensource.org [5] nace en 1990 como una iniciativa que está basada en el desarrollo de código fuente de software compartido, su principal característica es que el desarrollo de código es de una forma colaborativa, dicha iniciativa despertó el interés y la participación de muchas personas, principalmente en proyectos de Linux.

En si *Open Source* de acuerdo con [6], es un desarrollo informático que tiene una licencia que da acceso a cualquier código fuente para que los ingenieros puedan utilizar, escribir, modificar o redistribuir el código.

Como en todo sistema hay ventajas y desventajas del *Open Source.* Dentro de las ventajas que expone [6] podemos mencionar:

- Compartir, modificar y entender el código fuente del software
- Comunidades y foros para compartir soluciones de código
- Adaptación de software para necesidades específicas de las empresas
- Genera colaboración entre comunidades de programadores
- No inversión económica para comprar licencias
- Para el mantenimiento se puede contratar cualquier empresa o persona que tenga el conocimiento necesario
- Mayor continuidad del software ya que pertenece a la comunidad

Dentro de las desventajas que expone [6] podemos mencionar:

- No existe una oficina o un equipo de soporte técnico ya que toda la comunidad es la que está involucrada
- Si el proyecto de software no esta respaldado por alguna comunidad pude ser obsoleto o dejar de tener mejoras.

Después en 1998 nace OSI *open source initiative* como una organización, dentro de los cuales sus objetivos son la colaboración abierta a nivel mundial, el conocimiento generado y compartido entre empresas e Ingeniero y como objetivo principal la educación.

De acuerdo con [7] las licencias de Open Source son revisadas e integradas a una lista aprobadas por el OSD el cual por sus siglas en ingles se define como *The Open Source Definition* los términos de distribución para el software de código abierto debe cumplir con diferentes criterios y dependiendo de esos criterios se pueden aprobar y encontrarse en listas ya sea por nombres o categorías, dichas licencias registradas permiten ser usadas, modificadas y compartidas de manera libre.

Los términos dictados por la OSD y criterios a cumplir por [7] para registrar una licencia en OSI son los siguientes: Libre redistribución, Código Fuente, Trabajos Derivados, Integridad del Autor del código fuente, No discriminación hacia personas o grupos, No discriminación hacia cualquier campo de trabajo, Distribución de Licencia, La licencia no debe ser especifica de un producto, La licencia no debe tener restricciones hacia otro software y la Licencia debe ser tecnológicamente neutral.

Intel es una de las principales empresas que apoyan el open source, ya que estratégicamente genera una buena reputación ya que trabaja con empresas las cuales colaboran en grupo para la generación de nuevos productos de software, sin embargo, la presencia de Intel en el sector *open* source puede servir para atraer más clientes de segmento de mercado, en el cual Intel siempre ha estado presente.

StarlingX es un proyecto *open source* de Edge computing para tecnologías cloud por lo tanto su validación y todas las herramientas deben de tener la particularidad de también ser *open source,* ya que el propósito principal es que el proyecto sea colaborativo a nivel mundial.

Este proyecto se genero en diversas localidades en el mundo y en México se desarrolló la parte de integración, desarrollo y validación.

Este trabajo solo se enfoca en la parte de validación.

Dentro de los proyectos de software existen ciertas necesidades que se deben de cubrir, tales como la planeación de pruebas, el diseño de pruebas, la implementación de las pruebas, la ejecución de las pruebas y el mantenimiento de dichas pruebas, entre algunas otras actividades más. Sin embargo, todas ellas conforman la validación de un proyecto de software.

Hablar de validación es hablar de depuración de errores en un proyecto de software, por otro lado, si no existiera dicha validación de software por consecuente habría productos de software que simplemente no

funcionarían al cien por ciento, impactando directamente en la calidad de la empresa que lo genera y dañando directamente la reputación e imagen de dicha empresa, dando como resultado final: bajas ventas y usuarios y clientes no conformes. Dichas circunstancias hacen que la mayoría de las empresas de software generen proyectos de validación y fases de pruebas para sus productos de software.

Por concepto validación o pruebas de acuerdo con Tecnologia-Informacion [8], validación es un proceso que asegura la entrega de datos limpios y claros a los programas y aplicaciones y servicios que lo utilizan. Comprueba la integridad y la validez de los datos que se están introduciendo en diferente software y sus componentes. La validación de los datos garantiza que los datos cumplan con los requisitos y los parámetros de calidad.

La calidad del software es lo primordial para hacer una liberación de una versión de software en cualquier parte del mundo, la calidad asegura que el software tenga la mínima cantidad de errores y pueda ser ejecutado de manera optima por el usuario final

Por otro lado, otro concepto de validación de software como lo menciona Harbor Informática [9], es un proceso que demuestra a partir de documentos que el sistema cumple con las funciones con las cuales fue diseñado de acuerdo con las especificaciones de los requisitos del usuario y con la garantía de seguridad y trazabilidad de información.

Dentro de este proyecto de validación existen diferentes maneras de validar o probar el software.

Algunos tipos diferentes de validación o pruebas pueden ser:

- Validación/Pruebas de código
- Validación/Pruebas de tipo de datos
- Validación/Pruebas de rango de datos

- Validación/Pruebas de restricciones
- Validación/Pruebas estructurada

Generalmente en el proyecto de StarlingX se basó en validación de código mediante pruebas automatizadas

Las pruebas de software de acuerdo con SmartBear [10] puede ser funcionales y no funcionales.

Las pruebas funcionales se dividen en:

- Pruebas unitarias
- Pruebas de integración
- Pruebas de sistema
- Pruebas de aceptación

Las pruebas no-funcionales se dividen en:

- Pruebas de desempeño
- Pruebas de seguridad
- Pruebas de usabilidad
- Pruebas de compatibilidad

Para el proyecto de StarlingX se utilizaron pruebas de todos los tipos comenzaron a ser procesos manuales en los que se basaban las pruebas manuales y con el paso del tiempo fueron dieron paso a las pruebas automáticas, dichas pruebas automáticas fueron diseñadas con el propósito principal de reducir el tiempo de validación del proyecto y disminuir al mismo tiempo los errores humanos al momento de probar el software manualmente.

La funcionalidad principal de las pruebas de validación de software permite establecer un parámetro de calidad, mediante el cual se puede medir que tan bueno es el software creado o también para medir si esta listo el software para salir a producción o ser utilizado por el usuario final.

Dentro el mundo de las pruebas de software podríamos hablar de diferentes casos de éxito sin embargo uno de los sectores más críticos para las pruebas de software es el sector Medico, Financiero, Aéreo, Marítimo y Aéreo Espacial los cuales si no existiera una validación para generar calidad en el código entonces estaríamos hablando de diferentes catástrofes en dichos sectores a diario. Dicha validación de código ayuda a crear una calidad de código en el software y tener un nivel alto de confiabilidad donde el sistema sea lo mas seguro posible.

Empezar un proyecto de validación y tratar de implementar herramientas *open source* desde cero tiene ventajas.

Dentro de las principales ventajas se pueden mencionar la libertad de generar: diseño de procesos, selección de herramientas de validación, selección de reporteo de errores, seguimiento de errores, tipo de pruebas, cadencia de ejecución de pruebas, selección de metodologías para administrar el ciclo de desarrollo de software, entre otros.

También al momento de seleccionar alguna metodología o herramienta que pueda ayudar a el proyecto por ser tecnología *open source* se puede encontrar que también tienen ventajas y desventajas:

Ventajas:

- El poder seleccionar la herramienta que mejor se integre con las necesidades del proyecto
- No hay licencias en el área de open source
- Fácil acceso a repositorios
- Interacción rápida con la herramienta y el sistema, una vez que se descarga la herramienta se puede saber si se puede o no integrar al proyecto de StarlingX.

Desventajas:

- Es difícil hacer la investigación de la mejor herramienta puesto que hay muchos sistemas que proveen de buena funcionalidad.
- No siempre la mejor herramienta es *open source*
- Bajo soporte para herramientas *open source*
- Falta de documentación en herramientas *open source*

La validación de software juega un rol importante en el ciclo de desarrollo del software para pruebas y existen diferentes metodologías para poder llevar a cabo. Para el proyecto de validación de StarlingX se utilizo la metodología de *Waterfall* o cascada.

Robert Half International Inc. [11], menciona que dicha metodología es considerada la más vieja metodología estructurada del *Software Development Life Cycle (SDLC).* Sin embargo, también son de las metodologías que tienen un enfoque muy directo, esto se refiere a que son fases en cadena y no puede iniciar una fase si no ha terminado la fase previa a la misma, también menciona que es fácil de entender y administrar.

Software Development Life Cycle (SDLC) según [12], se refiere a una metodología claramente definida por procesos para crear software de alta calidad. La metodología SDLC se enfoca en las siguientes fases de desarrollo de software:

- Requerimientos de análisis

- Planeación

- Diseño de Software

- Desarrollo de software

- Pruebas

- Desarrollo

Utilizando la metodología *Waterfall SDLC* se generaron pruebas nuevas al principio manuales y después se automatizaban, dicha automatización se realiza con código de programación entre los cuales se encuentran bash, python y robot.

Una vez automatizadas se busca una herramienta la cual pueda administrar y ejecutar los diferentes tipos de pruebas a estas herramientas de programación se les llama *Framework de pruebas.*

Deacuerdo con [13], los *frameworks* de pruebas son guías que incluyen estándares de codificación, métodos de manejo de datos de prueba, procesos para guardar los resultados de pruebas entre otras cosas.

Una de las ventajas de utilizar un framework de pruebas es que usualmente los ingenieros al usar estructuras organizadas de codificación ahorran tiempo y son mas organizados al momento de crear casos de prueba para algún proyecto, esto impacta positivamente la calidad de cada caso de prueba y reduce el costo y el tiempo, variables de importantes de retorno de inversión que aseguran además la calidad de los casos de prueba generados para diferentes proyectos.

Los lenguajes de programación generalmente el lenguaje Python es utilizado en diversas aplicaciones de software, en la actualidad aplicaciones de todo tipo, son desarrolladas en python, esto se debe a su adaptabilidad a los diferentes sistemas en los que se puede ejecutar. Como lo dice Python Software Foundation [14] esta herramienta es fácil de aprender y un lenguaje de programación muy poderoso. Tiene estructura de datos altamente eficientes y un simple, pero muy efectivo trato para la programación orientada a objetos. Tiene una sintaxis elegantes y escritura dinámica, juntos con la naturalidad de su mismo interprete, lo hacen un lenguaje ideal para el

scripting y el desarrollo rápido de aplicaciones en muchas áreas en la mayoría de las plataformas. El intérprete de Python ha sido fácilmente ampliado con nuevas funciones y tipos de datos implementados en C y C++. Python es también un lenguaje muy adaptable para la customización de aplicaciones.

Así como lo dice la descripción del lenguaje, su simplicidad y efectividad fueron cualidades importantes para haber sido escogido al proyecto de validación de StarlingX junto con otros lenguajes, ya que su adaptabilidad de integración es muy buena.

Por otro lado, también se encuentran implementaciones de pruebas basadas en Bash, el cual como lo dice en su documentación de Free Software Foundation, Inc. [15], es el acrónimo usado para 'Bourne-Again Shell', el sobrenombre que tenia Stephen Bourne, el autor del ancestro directo del actual Shell de Unix sh, el cual apareció en la séptima versión de la edición de Bell Labs Research de Unix.

Bash es muy compatible con Shell y también incorpora componentes importantes desde el Korn Shell ksk y C Shell csh. Se pretendía que fuera una implementación de IEEE POSIX Shell and Tools parte de la especificación IEEE POSIX (IEEE Standard 1003.1). Ofrece mejoras en el funcionamiento sobre sh para ambos usos, interactivos y programables. Bash se puede ejecutar en la mayoría de las versiones de Unix y en algunos otros sistemas operativos.

Para el proyecto de StalirngX contaba en este punto con scripts de python y bash, sin embargo, aún había la necesidad de un administrador de pruebas o framework el cual a su vez mantuviera la suite de pruebas organizada.

Framework de Pruebas

Como buenas prácticas de software, una de las principales herramientas que necesitaba StarlingX para incrementar su calidad por medio de pruebas automatizadas era un framework de pruebas. Este framework de pruebas es una plataforma de software la cual permite automatizar pruebas manuales, generar ejecuciones automáticas, generar reportes y administrar diferentes tipos de pruebas como funcionales, stress, desempeño, sistema, integrar scripts con diferente lenguaje de programación entre otras cosas.

Dentro de los requerimientos para buscar esta herramienta open source, debería de contar con runners para poder ser ejecutado, fácil crecimiento de la suite que se genere, administración de suites por funcionalidad, componente y sistema, otro requisito de la herramienta era que estuviera basada en Python, ya que es más sencillo integrar pruebas de otros repositorios open source.

La razón por la cual utilizar un framework de pruebas es como lo dice la empresa S. Software [10] “incrementa la velocidad y la eficiencia de los equipos que trabajan en ese proyecto, mejora la calidad de las pruebas al mismo tiempo reduciendo los costos del mantenimiento y manteniendo los bajo los riesgos.”

De acuerdo con la empresa S. Software [10] las principales razones por las cuales se debe de implementar un proceso de automatización son:

- Mejora de la eficiencia de las pruebas
- Majo costo en mantenimiento
- Mínima intervención manual
- Cobertura máxima de pruebas
- Reusabilidad del código

Se investigaron diferentes frameworks tales como: Selenium y Robot.

Selenium era una buena opción en un inicio ya que se necesitaba generar pruebas para el componente de horizon, el cual es el componente donde se encuentran la interfaz web de usuario, sin embargo, la mayoría de los componentes no son web, por lo tanto, no se podía integrar completamente al proyecto de validación como para ser la herramienta principal de validación.

Robot fue la opción que se tomó en ese momento, por que cubría la mayoría de los requerimientos que necesitábamos para integrar una herramienta de validación, dichas razones fueron:

- Herramienta *Open Source*
- Fácil integración con diferentes lenguajes
- Fácil interacción con Selenium Web Driver el cual nos ayudaba al mismo tiempo poder validar las funciones de la web
- Herramienta basada en Python
- Fácil de entender, configurar y usar
- Uso de palabras clave las cuales se utilizan como funciones y son fáciles de crear y utilizar

De acuerdo con la documentación de Robot Framework Foundation [16], robot es una framework genérico open source de automatización fue creado originalmente para pruebas de aceptación y procesos robóticos de automatización. Es simple texto plano y puede fácilmente extenderse con librerías implementadas usando python o java.

Herramientas de validación

No solo el framework de pruebas es suficiente para automatizar pruebas para este proyecto, también hay herramientas necesarias para integrar dichas pruebas en un solo lugar. Esto es necesario para tener un solo origen estructurado en donde se puedan crear, eliminar y modificar las pruebas, para esto existen los repositorios como Gitlab o Github, los cuales su función principal es poder exponer proyectos de forma privada o publica y por otro lado proyectos colaborativos.

Por otro lado, al momento de automatizar las pruebas, se pueden generar rutinas de instalación de dicho reportorio de pruebas en cualquier parte lo cual lo hace un sistema de validación adaptable y escalable.

En un inicio de utilizo GitHub sin embargo Intel decidido que se cambiara a gitlab por mayor seguridad dentro de la misma red de Intel hasta que el producto estuviera maduro y poder exponerlo poco a poco a los clientes.

Gitlab [17], menciona que es una empresa de núcleo abierto la cual desarrolla software para ser usado por el SDLC *Software Development Life Cycle* por mas de 100,000 organizaciones, con aproximadamente 30 millones de usuarios registrados. La misión de Gitlab es cambiar todo el trabajo de solo lectura a lectura y escritura, en donde todos puedan contribuir, consumidores convirtiéndose en contribuidores para generar e incrementar el progreso de la humanidad.

Su misión también esta pasada en el mundo del *open source* por lo cual Intel decide que el proyecto resida en esta herramienta.

Dicha herramienta específicamente es una aplicación basada en web que forma parte generalmente del proceso de DevOps aparte de funcionar y proveer un repositorio git también funciona como versionador de software,

administrador de errores y generador de cargas de trabajo llamados *pipelines* los cuales son secuencias de actividades automatizadas

Por definición DevOps como lo menciona AWS [18], es una combinación de filosofías culturales, prácticas y herramientas que incrementan las habilidades de las organizaciones para entregar aplicaciones y servicios a una alta velocidad, evolucionando y mejorando productos más rápido que las organizaciones que utilizan infraestructura de desarrollo de software tradicionales. Dicha velocidad habilita a las organizaciones para trabajar con sus clientes de una forma efectiva y ser mas competitivos en el mercado.

Dicho esto, dentro del ambiente que se creó de DevOps dentro de StarlingX, una vez generado un repositorio de pruebas, en el proyecto de validación de StarlingX se vio la necesidad de implementar un autómata de procesos de pruebas.

Jenkins como lo menciona su documentación [19] es un servidor autocontenido open source el cual puede usarse para automatizar muchas actividades relacionadas con la construcción, pruebas, entregas o desarrollo de software. Jenkins puede ser instalado a través de sus sistemas de paquetes nativos, Docker o inclusive una aplicación única ejecutada en una maquina que tenga Java Runtime Environment (JRE) instalado.

Con esta herramienta se automatizaron los procesos para generar procesos autónomos o dependientes de validación de pruebas. De esta forma se pudieron implementar diferente tipo de actividades o secuencia de procesos con los cuales asegurábamos un proceso automatizado de validación con el objetivo de encontrar errores de una manera rápida, eficaz y automatizada.

Dichas ventajas fueron los objetivos principales al inicio del proyecto y en base a esos objetivos se eligió implementar procesos de DevOps ya que

sus beneficios tales como: Velocidad, Entrega rápida, Confianza, Escalabilidad, Mejora en la colaboración y seguridad, como lo describe AWS [18] son beneficios que al final generan un producto de mayor calidad y alta confiabilidad, ingredientes que tienen un peso importante dentro de la industria del software.

CAPITULO II

3. DESCRIPCIÓN DEL PROYECTO

2.1 ANTECEDENTES DEL PROYECTO

Dentro del mundo del *Open Source* para Intel había la necesidad de generar competencia en el mercado en el segmento *cloud*, específicamente en sistemas *Edge Cloud*, los cuales son sistemas *cloud* industriales de baja latencia. La apuesta de Intel por un sistema *edge cloud* completamente *Open Source* listo para implementarse es el objetivo principal de Intel para sus clientes que utilicen tecnologías abiertas.

Dicho sistema estaba proyectado para industrias automotrices, medicas y de manufactura, en donde dicho sistema seria instalado en las fábricas de los clientes con soporte técnico de Intel y de la comunidad Open Source, dichos sistemas *edge cloud* al momento de ser instalados en las fábricas trabajara de forma rápida y eficaz, pero, sobre todo, el objetivo principal es el generar sistemas de baja latencia.

Debido a la naturaleza del proyecto, la validación de todos los componentes cloud, tanto su desarrollo desde cero, como su integración y su funcionamiento tienen que ser previamente validados antes de ser instalados en con el cliente, dichas circunstancias generaron una nueva tarea, la cual fue crear un proyecto de validación para este sistema *edge cloud.*

En los inicios del proyecto debido a las diferentes empresas con las cuales Intel empezaba a colaborar el sistema iba cambiando de nombre constantemente.

Al inicio, Intel empezó a colaborar en dicho proyecto con la empresa *WindRiver* lo que llevo a la necesidad de homologar procesos de diferentes empresas.

Homologar procesos significaban cambios y dichos cambios sucedieron de nuevo cuando WindRiver dejo el proyecto y solo quedo Intel apoyado ahora por *Open Stack*, llevando el proyecto a cambiar de nombre dos veces. En dichos cambios al proyecto se le denomino, Akraino y después Madawaska.

El proyecto fue cambiando de diseño y logos hasta que se asentó con solo Open Stack e Intel de esta forma surgió StarlingX.

Open Stack es una plataforma de software *open source* para computo en la nube, desarrollado principalmente como infraestructura como servicio (IaaS) para diferentes tipos de nube, como privada y pública. Intel se unió a colaborar en este proyecto basado en Open Stack para generar presencia dentro de las plataformas open source para la nube tipo *edge*.

StarlingX es un sistema cloud flexible el cual permite ejecutar contenedores de servicios, máquinas virtuales y/o desarrollar ambientes *bare metal*. Este sistema es una solución optimizada para servicios de cloud para soluciones de Edge Computing.

La característica e importancia por la cual Intel aposto en tecnologías de Edge Computing, es porque dicha tecnología se enfoca principalmente a la baja latencia, esto es generado debido a la interacción de los dispositivos con los recursos de cómputo y almacenamiento de datos los cuales se encuentras más cerca de los dispositivos que generan datos y esto permite que la transferencia de información sea más rápida y con menos tiempos de espera.

Intel Guadalajara en México fue el candidato para desarrollar la validación del código de StarlingX, en paralelo Canadá, USA y China también colaboraron.

Los componentes que conforman StarlingX son: redes, almacenamiento, identidad, imagen, almacenamiento de objetos, instancias, visualizadores, entre otros.

En el inicio del proyecto había dos objetivos, el primer objetivo por el cual empezó este proyecto es el de implementar un sistema validación robusto y automatizado para los componentes que integran StarlingX.

El segundo objetivo, sería el de buscar en un futuro con una base sólida de validación, automatización en ambientes virtuales y bare metal, ser una herramienta base en un sistema de integración continua para Intel, la cual el principal objetivo es aportar soluciones de servicio a la comunidad de Open Source.

La problemática que vio Intel en el mercado fue implementar StarlingX para soluciones globales y abiertas en donde la solución sea competitiva, de tal manera que tenía que implementar un sistema de baja latencia, modular, seguro y altamente escalable para poder incursionar en los mercados de manufactura, automóviles, etc.

En un inicio se observaron dos desventajas que se trataron de solucionar con este proyecto de validación de StarlingX.

Primero la calidad del código de StarlingX, siendo un sistema basado en Open Stack y con varias empresas colaborando juntas, era difícil de controlar, verificar y validar cambios y nuevo código integrado, sin embargo, el proyecto de validación ayudaría a solucionar y reportar errores a Open Stack del código que integro el mismo Open Stack.

Por otro lado el segundo problema a combatir seria encontrar errores en el mismo desarrollo de StarlingX que se esté haciendo por parte de Intel,

sin embargo el proyecto de validación también ayudaría y apoyaría de manera importante la confiabilidad del producto para ser implementado en cualquier industria que necesite sistemas cloud tipo *edge*, el trabajo medular se concentró en combatir y encontrar la mayor cantidad de bugs existentes dentro del código de StralirngX mediante el proyecto de validación.

Para el proyecto de validación existieron una combinación de metodologías. Se utilizaron dos diferentes metodologías tanto para el desarrollo de código en robot en el cual se validaba las pruebas como para el proyecto en general.

Para tareas de configuración, compras e infraestructura se utilizó y se controló con un tablero de Kanban, para el desarrollo de código de las pruebas se utilizó la metodología en cascada, para las tres diferentes etapas en que se trabajaría: Integración de pruebas existentes en repositorios externos open source, creación de nuevas pruebas manuales y la automatización de pruebas

Justificación:

Dentro del proyecto StarlingX se buscó mejorar su calidad de su código fuente, siendo este un proyecto Open source donde interactuaban diferentes empresas, el desempeño del código no era el óptimo puesto que no había homologación de buenas prácticas de código, la validación era pobre o casi nula, ya que era un proyecto nuevo en el cual solo se integraba el código mas no se validaba a profundidad, generando errores, es por eso que para cubrir esas necesidades se necesitaba implementar un servicio de validación para el StarlingX, dicho proyecto de validación ayudaría a mejorar la calidad del código, ya que mediante la validación de código automatizada y el hallazgo oportuno de errores, cubriría de inicio una gran parte del objetivo con el que se planteó este proyecto, sin embargo el tener un proyecto más ordenado serviría de medio y fomentaría la colaboración con la comunidad y

el uso del software para empresas, de esta forma Intel estaría incursionando en tecnologías de cloud tipo edge para la comunidad Open Source

Alcance del proyecto:

1. Generar un servicio de validación y pruebas básicas de sanidad para el proyecto de StarlingX mediante el *framework* de Robot
2. Integración de un framework de pruebas
3. Desarrollo de planes de pruebas para los componentes
4. Creación de un repositorio de pruebas
5. Generación de casos de prueba para los diferentes componentes
6. Generar la documentación de las pruebas de validación
7. Creación de una suite de pruebas de sanidad, funcional y sistema
8. Integración de un manejador de revisiones de código
9. Integración de una herramienta de integración continua como Jenkins para generar rutinas automáticas de validación de código y componentes

2.2 Objetivo del Proyecto

El objetivo es generar un servicio de validación para el proyecto de *edge computing* de StarlingX y todos sus componentes, en el cual, se validaría todos sus componentes mediante un framework de pruebas, integrando código nuevo o integrando código existente de otras fuentes Open source a un nuevo repositorio de pruebas, dichas pruebas estarían enfocadas en encontrar errores en el código fuente, proporcionando mayor calidad al código existente y la mismo generando una base de validación robusta para ser un punto de referencia y ofrecer este servicio de validación y colaboración a la comunidad Open Source.

2.3 DESCRIPCIÓN METODOLÓGICA

En el proyecto de validación para StarlingX existieron metodologías y herramientas que ayudaron a administración del proyecto.

Dentro de Intel en el área de *Open Source* la herramienta de Kanban es una herramienta muy utilizada con gran aceptación también en la comunidad *Open Source,* por esta simple razón se decidió implementarla.

Por otro lado, para dichas actividades que no eran propias del proyecto de Validación también se implementó el mismo método para gestionarlas mediante el cual también fue Kanban, estas actividades que no eran de pruebas o validación de código se controlaban con un tablero Kanban, la benevolencia del Kanban en estas tareas ayudaba a repartir la carga de trabajo entre los Ingenieros y poder llevar un mejor control y seguimiento de las actividades.

Las actividades que se administraban con el Kanban propiamente del proyecto de validación eran:

Compra, instalación y configuración de infraestructura, para estas actividades la administración, fechas de compromiso y entrega eran cruciales para el proyecto, con Kanban se podía administrar y visualizar a corto y mediano plazo, el crecimiento de la infraestructura.

Devops, para estas actividades era de mucha ayuda el Kanban ya que se trabajaba con equipos diferentes como integración, desarrollo y devops.

Kanban ayudaba a determinar y delimitar las actividades de cada equipo, así como sus requerimientos y entregables.

Compra de insumos para el proyecto, debido a la naturaleza del proyecto había la necesidad de comprar dispositivos periféricos o de almacenamiento externo, por lo tanto, se necesitaba utilizar una herramienta

para cubrir dichas actividades y llevar un control sobre lo que se gastaba para tener un mejor control de presupuesto y visibilidad del proyecto.

Dicha metodología la cual tuvo su origen en los procesos de producción de Toyota, en la que se utilizan tableros o tarjetas para ayudar a identificar las necesidades a cubrir en el proyecto, ayudo en tener todas las actividades concentradas y de ahí, partir con el monitoreo de actividades, actividad por actividad. Por otro lado, dentro de la generación de código para las pruebas en robot se utilizó la metodología de cascada.

Dentro del desarrollo del proyecto y sus entregables dicha metodología se iba adaptando cada vez utilizando etapas diferentes de la misma metodología, dichas etapas se integraban o se descartaban dependiendo como lo requería el proyecto en ese momento y dependiendo también de los componentes nuevos que se iban integrando al paso del tiempo y las modificaciones que se hacían a la suite de pruebas.

El proyecto de validación, como su nombre lo indica, se trata de probar/validar todos los componentes de un sistema *cloud* de código abierto.

Dicho esto, el requerimiento principal era el generar casos de prueba que cubrieran de forma funcional y sistemática las principales funciones del proyecto cloud, generando código nuevo para casos de prueba no existente en la comunidad *Open Source* o en el caso de existir dicho código, integrar el código existente y portarlo para su funcionamiento correcto con tecnologías de Intel.

Dentro del desarrollo de código de las pruebas en donde se implementó la metodología de cascada, se contemplaron las siguientes etapas: Requerimientos, Planeación, Diseño, Desarrollo, Validación y Mantenimiento.

- Requerimientos

Dentro de la etapa de requerimientos se obtiene la información con la cual se implementa el desarrollo de código para los nuevos casos de prueba que se integran para los componentes.

Dichas actividades requerían información para poder generar los requerimientos del diseño de las pruebas como también, herramientas para mejorar la eficiencia en los procesos de validación entre otros datos. Para la obtención de estos datos se trabajó en diferentes actividades tales como:

- Análisis de Datos, para recabar la información de cómo integrar todas las partes del proyecto y sus herramientas
- Juntas y *brainstorming*, para generar los acuerdos de los procesos en base a como funcionara el proyecto
- Juntas con Arquitectos del proyecto para alinear el proyecto a los estándares y herramientas
- Juntas con Desarrolladores del proyecto para entender los requerimientos de validación y la importancia exacta en que o cual funcionalidad de algún componente del código fuente de StarlingX
- Cursos, los cuales se implementaron con personas que ya tenían habilidades en OpenStack y permitiera una rápida curva de aprendizaje en el sistema cloud de StarlingX

- Planeación

En la etapa de planeación, usualmente se generaban planes de acción para todas las actividades del proyecto de validación, como los alcances de las pruebas, tipos de pruebas, planes de pruebas, estrategias de pruebas. Por otro lado, la planeación también ayudaba a anticiparse a posibles errores en la producción de pruebas o al

entregar una nueva versión de resultados para alguna versión del proyecto StarlingX. Por otro lado, se planeaban los criterios de salida para las pruebas para la creación de nuevas pruebas

- Diseño

Dentro del diseño de los casos de prueba, básicamente había tres tipos de técnicas para la obtención de información, la primera seria mediante la investigación desde cero de un componente, entender los requerimientos para diseñar el caso de prueba y generar el caso de prueba con código nuevo desde cero.

La segunda es la adecuación de casos de pruebas de los colaboradores anteriores con el proyecto StarlingX, adaptar y mejorar esos casos de pruebas con la infraestructura de hardware que se utiliza

La tercera técnica es la adecuación de casos de prueba de otros proyectos de cloud, los cuales se pueden generar casos de pruebas basados en esas validaciones.

- Desarrollo

Dentro de la etapa de desarrollo, los casos de prueba que se crean y se integrarán al repositorio máster de pruebas, previamente validados por los miembros del equipo o cada dueño del componente que hace el desarrollo, por otro lado, también es validado con la suite para saber si finalmente puede formar parte del repositorio oficial de validación. Dentro de robot con el tiempo se fueron implementando diferentes suites, las cuales implementaban diferentes planes de validación tales como:

Sanity, en esta etapa se desarrollaban pruebas de sanidad, las cuales serían la mínima cantidad de pruebas a ejecutar para asegurar que el sistema esta funcional y sin errores en sus componentes más importantes.

Funcional, en este plan de validación se encuentras las pruebas funcionales, las cuales son desarrolladas para validar y asegurar el funcionamiento correcto de las principales funciones de un componente ya sea: *network*, almacenamiento, etc.

Stress, el objetivo de estas pruebas de validación es llevar el sistema a un nivel de funcionamiento tal, que pudiera llegar al punto del fallo. En este plan de validación se encuentras las pruebas que generan stress a los componentes principales del sistema, dicho stress simula las cargas excesivas y reiterativas de trabajo para asegurar que los componentes trabajen de una forma eficiente, simulando el desgaste de los componentes, lógica del código y calidad de la infraestructura a través del tiempo.

Sistema, este plan de validación se genera con diferentes pruebas de validación y no solo de un solo componente, la mayor característica es que se integra por medio de diferentes pruebas de diferentes componentes y al final se prueba como un sistema completo, esto con la finalidad de ver como interactúan los componentes entre sí y su desempeño.

Regresión, dentro de este plan existe las pruebas de validación con las cuales se asegura de cubrir un cierto segmento del proyecto o del componente que tuvo un cambio recientemente, ya sea por un parche de código o una integración de nuevas funcionalidades o

componentes. Dichas pruebas validan que los nuevos cambios no afecten partes del proyecto o componentes.

- Validación

En la etapa de validación es donde se hace la ejecución de las pruebas nuevas creadas por Intel y las pruebas integradas y portadas de la comunidad a la suite oficial de validación.

En esta etapa también se utiliza para verificar si los nuevos casos de prueba no tienen algún conflicto con el resto de la suite o si las pruebas existentes de la comunidad que fueron integradas y portadas para trabajar con tecnologías de Intel funcionan tal cual se diseñaron.

Dentro de esta etapa también se generan diferentes tipos de validaciones, como *smoke*, regresión, funcional, sistema, sanidad entre otros.

Smoke testing hace referencia a la validación a nivel superficial de errores en el código de un componente o de un sistema después de haber integrado la resolución en el código fuente.

Regresión hace referencia a la validación a nivel profundo de errores en el código de un componente o de un sistema después de haber integrado la resolución en el código fuente.

Funcional hace referencia a la validación de las principales funciones de un componente o sistema.

Sanidad hace referencia a la validación de los principales componentes que integran el sistema para verificar que su funcionamiento básico es correcto.

- Mantenimiento.

  En la etapa de mantenimiento se busca generar una suite estable, robusta e inteligente que se adecue a las necesidades del proyecto por lo tanto la integración de nuevo código, modificación de código y eliminación de código usualmente se realiza con el tiempo en esta etapa. Estas modificaciones usualmente son requeridas para mejorar el desempeño de la suite de pruebas y al mismo tiempo mejorando rendimiento del hardware.

2.4 PLANEACIÓN CRONOLÓGICA DEL PROYECTO

Fases de la Planeación:

1. Obtención de información del proyecto a validar
2. Planeación de la curva de aprendizaje en el software, hardware y focos de validación para el proyecto
3. Recopilación de información de validación externa por (Windriver, OpenStack)
4. Documentar la información recabada en un repositorio de Intel
5. Integrar, documentar y ejecutar pruebas existentes manuales
6. Integración del framework de pruebas (Robot)
7. Adaptación de pruebas y automatización en el Framework (Robot)
8. Integración de CI/CD

2.5 Descripción de las Actividades

Las actividades en el proyecto de validación se concentraron en el modelo en cascada convencional. Desde el inicio del proyecto ya había ciertos requerimientos que se tenían que cubrir en hardware y software.

Para la infraestructura se tenía que generar centro de datos, configurar *switches* y servidores para poder ser utilizados al momento de generar un *clúster* de StarlingX. Del lado de software se generaban imágenes con el software preinstalado de StarlingX para ser utilizado por máquinas virtuales, servidores o *clúster* de bare metal.

Al inicio del proyecto era necesario generar un ambiente de pruebas, esto significaba crear un repositorio para las pruebas de StarlingX las cuales en un inicio tendrían que ser manuales y después automáticas, por lo cual se tuvo que decidir el framework de pruebas, con lo que se decidió que fuera Robot.

Después de que la infraestructura de hardware estuviera según los requerimientos, la siguiente etapa sería el desarrollo de las pruebas para la versión de software correspondiente al modelo de cascada.

El objetivo era desarrollar pruebas y validar el proyecto de StarlingX como tal y los nuevos desarrollos de integración que se requieran en un futuro, dichas actividades aseguran la calidad del código de StarlingX.

Dentro de estas actividades existieron tres fases: integración de pruebas existentes en repositorios externos, creación de pruebas manuales y creación de pruebas automatizadas.

Fase de integración de pruebas existentes en repositorios externos

Para la integración de pruebas existentes en repositorios externos las actividades eran las siguientes:

- Requerimientos
    - Recabar pruebas de repositorios globales que puedan ser integradas en StarlingX
    - Utilizar pruebas en el área de Open Source
    - Documentar pruebas integradas
    - Integrar pruebas solo las correspondientes específicamente al componente a validar
    - Integrar pruebas específicamente a la función a validar

- Planeación

    - Planeación de funcionalidades a probar por componente
    - Planear búsquedas en repositorios externos de validación
    - Planear que tipo de pruebas se pretenden integrar
    - Planeación del tiempo a integrar dichas pruebas existentes
    - Planeación de qué clase de código integrar
    - Planeación de procesos para integrar pruebas existentes en otros repositorios

- Análisis

    - Análisis de los requerimientos del proyecto con el equipo de desarrollo y validación
    - Análisis de las herramientas disponibles Open Source para poder validar el proyecto de acuerdo con los requerimientos

- Análisis de presupuesto para disponer de recursos tecnológicos y humanos
- Análisis de los requerimientos de hardware y software a utilizar
- Análisis de los procesos a crear
- Análisis e identificación temprana de cuellos de botella
- Análisis y clasificación de prueba

- Integración

  - Entender el código
  - Eliminar funciones innecesarias
  - Modificar las pruebas en base a nuestra infraestructura
  - Integrar solo funciones que ejerciten la funcionalidad del requerimiento
  - Utilizar buenas prácticas de codificación

- Diseño

  - Diseñar las pruebas nuevas con el código externo en el template del proyecto de validación
  - Diseñar el código de acuerdo con las buenas prácticas para desarrollar código
  - Diseñar el código de acuerdo con las prácticas de documentación del proyecto de validación de StarlingX

- Validación

  - Probar lógica de la prueba integrada
  - Validar que contenga el template de pruebas correcto

- Validar las buenas prácticas de programación para el desarrollo de la prueba Case
- Validar que la prueba valide la función para la cual fue creado
- Validar que no afecta a otras pruebas no relacionadas con la funcionalidad de la prueba
- Validar que la prueba no entregue falsos positivos

- Mantenimiento

  - Integrar más pruebas a la suite por componente
  - Buscar nuevas pruebas en repositorios externos
  - Resolver errores de dichas pruebas
  - Agregar soporte para versiones nuevas de StarlingX
  - Mejorar la lógica y reducir código
  - Simplificar código mediante funciones globales
  - Crear y reportar errores en la suite
  - Crear y reportar errores del código de StarlingX
  - Dar seguimiento a los errores junto con el equipo de desarrollo
  - Mantener los Jiras actualizados

Fase de creación de pruebas manuales:

- Requerimientos
  - Crear pruebas manuales por función
  - Clasificar pruebas manuales creadas dentro del repositorio
  - Documentar los pasos para crear la prueba
  - Pasos de ejecución de prueba siempre actualizados

- Planeación
  - Planeación de funcionalidades a probar por componente
  - Planeación del tiempo en estudiar el componente
  - Planear que tipo de pruebas se pretenden integrar
  - Planeación del tiempo a crear desde cero dichas pruebas existentes
  - Planeación del repositorio de pruebas manuales
  - Planeación de procesos para pruebas nuevas
  - Planeación de documentación de pruebas y resultados manuales

- Análisis
  - Análisis de los requerimientos del proyecto con el equipo de desarrollo y validación
  - Análisis de las funciones a integrar
  - Análisis de presupuesto para disponer de recursos tecnológicos y humanos
  - Análisis de los requerimientos de hardware y software a utilizar
  - Análisis de los procesos a crear para las pruebas manuales
  - Análisis e identificación temprana de cuellos de botella
  - Análisis y clasificación de prueba
  - Análisis del tiempo que toma en ejecutar la prueba una vez diseñada

- Validación

  - Probar la secuencia de pasos para lograr ejercitar la funcionalidad de cada requerimiento
  - Probar la funcionalidad que involucra solo el requerimiento

- Validar que todos los prerrequisitos sean cumplidos para ejecutar una prueba
- Validar que se esté validando el Hardware y la infraestructura correcta
- Validar que se prueben las versiones más recientes de software involucradas
- Validar si presenta algún tipo de error la función que se está validando

- Mantenimiento

  - Mantener los pasos de las pruebas actualizados
  - Mantener los planes de validación actualizados con los resultados
  - Incrementar nuevas pruebas en los diferentes componentes
  - Validar errores mediante las pruebas manuales
  - Crear y reportar errores del código de StarlingX
  - Dar seguimiento a los errores junto con el equipo de desarrollo
  - Mantener los Jiras actualizados

Fase de creación de pruebas automatizadas

- Requerimientos
  - Automatizar todas las pruebas manuales
  - Utilizar un repositorio de pruebas automatizadas
  - Integración completa de pruebas automatizadas a procesos de CI/CD
  - Automatizar pruebas sin afectar la ejecución de ninguna suite
  - Clasificar las pruebas automatizadas dentro del componente correspondiente.

- Planeación

    - Planeación de funcionalidades a probar por componente
    - Planeación del tiempo en desarrollar la prueba o pruebas automatizadas para el componente
    - Planear que tipo de pruebas se pretenden automatizar
    - Planeación del repositorio de pruebas automatizadas
    - Planeación de procesos para automatizar pruebas nuevas
    - Planeación de documentación de pruebas y reporte de resultados automatizado

- Análisis

    - Análisis de los requerimientos del proyecto con el equipo de desarrollo y validación
    - Análisis de las funciones a automatizar
    - Análisis de presupuesto para disponer de recursos tecnológicos y humanos
    - Análisis de los requerimientos de hardware y software a utilizar
    - Análisis de los procesos a implementar en la automatización
    - Análisis e identificación temprana de cuellos de botella
    - Análisis y clasificación de pruebas
    - Análisis del tiempo que toma en ejecutar la prueba una vez automatizada
    - Análisis de indicadores de desempeño

- Diseño
    - Diseñar las pruebas nuevas con el template del proyecto de validación
    - Diseñar el código de acuerdo con las buenas prácticas de diseño

- Diseñar la documentación del código de acuerdo con las prácticas de documentación del proyecto de validación de StarlingX
- Diseñar los procesos para CI/CD
- Diseñar los flujos de operación de la infraestructura automatizada
- Diseñar los reportes automáticos para las pruebas

- Desarrollo
  - Desarrollar las pruebas con el template del proyecto de validación
  - Desarrollar la prueba de acuerdo con las buenas prácticas de codificación
  - Desarrollar la prueba dentro de la suite correspondiente
  - Desarrollar los procesos de CI-CD con los flujos de operación
  - Desarrollar e implementar los reportes automáticos

- Validación

  - Probar la secuencia de pasos automatizados para lograr ejercitar la funcionalidad de cada requerimiento
  - Probar la funcionalidad que involucra solo el requerimiento
  - Validar que todos los prerrequisitos sean cumplidos para ejecutar una prueba
  - Validar que se esté probando el Hardware y la infraestructura correcta de acuerdo con los requerimientos de las pruebas
  - Validar que se prueben las versiones más recientes de software involucradas
  - Validar si presenta algún tipo de error la función que se está validando

- Mantenimiento

    - Mantener la suite y repos de pruebas automatizados al día
    - Mantener los planes de validación actualizados con los resultados
    - Incrementar nuevas pruebas en los diferentes componentes
    - Validar errores mediante las pruebas automatizadas
    - Crear y reportar errores en la suite
    - Crear y reportar errores del código de StarlingX
    - Dar seguimiento a los errores junto con el equipo de desarrollo
- Mantener los Jiras actualizados

2.6 RESUMEN DE LA DOCUMENTACIÓN E INFORMACIÓN RECABADA

En el proyecto de validación de StarlingX se generó documentación de diferentes tipos de acceso, tales como confidenciales y no confidenciales, cabe mencionar que la mayoría de la documentación del proyecto de StarlingX no se encuentra en Guadalajara ya que el proyecto como tal se creo en USA, para el proyecto de validación el cual se generó en México, la documentación se resume a:

- Planes de Pruebas
- Formatos para Pruebas nuevas
- Documentación de Innovación de nuevos procesos tales como: Validación como servicio VaaS
- Repositorios actuales de StarlingX
- Links hacia Documentación oficial de StarlingX

Una de las situaciones desfavorables para mostrar documentación sobre el proyecto de validación de StarlingX es que las políticas de Intel cambiaron y no se renovó la licencia de Google Drive en donde residían todos los documentos desde su inicio y al momento ya no se puede acceder a la mayoría de la información.

2.7 RESULTADOS OBTENIDOS EN EL PROYECTO REPORTADO

Dentro de los resultados obtenidos para el proyecto de validación de StarlingX se pueden mencionar varios resultados positivos, los cuales involucran procesos nuevos diseñados e implementados para el proyecto de validación, nueva infraestructura configurada e instalada en el laboratorio de Intel, investigación de tecnologías relacionadas con tecnologías *cloud,* integración de lenguajes, integración de sistemas para la generación de entrega e integración continua, documentación y repositorios de pruebas e incremento de un 60% de recursos humanos durante la vida del proyecto de validación.

Dentro de los detalles de los resultados obtenidos podemos decir que se investigaron de 7 nuevas tecnologías relacionadas con tecnologías cloud para entender su funcionalidad y poder generar pruebas que puedan cubrir su funcionalidad, dichas pruebas fueron directamente para los componentes de openstack como neutron, glance, nova, cinder, ceilometer, keystone, horizon.

Integración de 3 lenguajes tales como: Python, Bash y Robot, también, por otra parte, es importante mencionar la generación de contenido de prueba manuales, su documentación y automatización. Dentro de la división de la suite de pruebas se encuentran divisiones de pruebas de ambiente de configuración, las pruebas de sanidad y pruebas por componente, generando un total aproximado en ese momento de 850 pruebas automatizadas. Hoy en día es difícil calcular y obtener el número exacto que en ese momento se generó de pruebas ya que el numero va incrementado al paso del tiempo, ya que el proyecto sigue en la localidad de Intel Alemania.

8 nuevos Procesos creados e implementados los cuales son: planeación de pruebas, diseño de pruebas, documentación, configuración de ambientes bare metal y virtuales, instalación de herramientas de validación, integración

de ambientes de pruebas, ejecución y reporteo. Dichos procesos mejoraron los tiempos de validación y la calidad de las pruebas, lo cual impactaba positivamente en el proyecto ya que generaba mayor confiabilidad y confianza en el código, lo que permitiría a Intel ganarse un puesto importante dentro de la comunidad Open Source.

2 nuevas propuestas de mejora que no fueron implementadas, tal es el caso de *Validation as a Service* y *Testing as a Service*, esta ultima propuesta fue generada para modular aun mas los servicios de validación del proyecto de StarlingX.

4 repositorios en los cuales residían la suite de pruebas y documentación, pruebas de experimentación y pruebas de configuración de sistema. De esta forma se tenia un mayor control y administración en la documentación y pruebas, ya que cada segmento contaba con su propio repositorio, de esta forma la organización de integración de contenido y mantenimiento era mas eficaz y de mayor calidad.

Capítulo 4 - Conclusiones.

4. CONCLUSIONES

4.1 LECCIONES APRENDIDAS

Inicialización y puesta en marcha de proyectos desde cero:

Al momento de generar proyectos desde cero, todo el proceso de inicialización del proyecto y puesta en marcha en su totalidad todo el conocimiento nuevo que genera, problemas y solución en los procesos de infraestructura y procesos de validación fueron lecciones aprendidas las cuales crearon que el equipo de Guadalajara creciera, ganara confianza y en base al conocimiento adquirido pudiera implementar de manera correcta y eficiente las operaciones a nivel validación para el código de StarlingX

Implementación de Metodologías:

El proyecto de validación de StarlingX fue muy retador, sin embargo, se aprendieron procesos tales como las etapas de las metodologías de *waterfall,* dichas fases se utilizaron para la planeación, diseño, desarrollo y ejecución de las pruebas. Con esta metodología fue mas sencillo administrar el proyecto y entender los gaps de validación para en un futuro diseñar planes de validación con los cuales se iban expandiendo y perfeccionando con el tiempo.

Manejo de herramientas, procesos y lenguajes de pruebas, implementación de procesos de CI/CD y DevOps:

Por otro lado, todas las herramientas de validación que se utilizaron, tales como: lenguajes de programación como python, bash y robot, fueron aprendidos y aplicados al proyecto. Muchos colaboradores están muy satisfechos puesto que este conocimiento es muy valioso para sus posiciones actuales.

Innovación:

También se conocieron diferentes procesos para la solución de problemas en la instalación de diferentes herramientas, dichos procesos dentro de Intel se le conocen como *“BKM” Best Knowledge Method* por sus iniciales en inglés.

Dichos procesos ayudaban a nuevas personas que se integraban al equipo de validacion al paso del tiempo, también como apoyo a los colaboradores internos para reproducir alguna configuración o simplemente como una ayuda.

Al momento de integrar diferentes herramientas y automatizarlas no podemos dejar de lado los procesos de automatización de tareas y herramientas utilizadas dentro de DevOps como Jenkins. La integracion y automatización de los *pipelines* de Jenkins fueron vitales para la automatización completa del proyecto de validacion de StarlingX. Estas configuraciones y nuevos procesos fueron de gran ayuda y de mucho conocimiento para potencializar las habilidades de cada contruibuidor en el equipo de validación.

El proyecto de validación como servicio (VaaS) fue una innovación que, si bien no fue implementada en su totalidad, el conocimiento que genero en la planeación del proyecto (VaaS) fueron gratas lecciones aprendidas.

4.2 PROPUESTA DE MEJORA

StarlingX es una propuesta de Proyecto sobre la innovación en los servicios en el área del Open source el cual desarrolla tecnologías cloud para la comunidad global, el proyecto de validación iniciado en Guadalajara, México provee diseño y creación, validación, reporteo de errores y mantenimiento de código sin embargo los procesos están por separados, tienen dependencias de recursos y sistemas y solo hay un finito número de

recursos. Por otro lado, como la validación la provee Intel el proceso solo es interno y la forma de trabajo con la comunidad es bajo peticiones de colaboradores de cualquier parte del mundo por medio de mails, los cuales entran en una pila de trabajo y se van atendiendo conforme los recursos (humanos y de sistema) estén disponibles, lo cual en un futuro no sería una solución escalable ni tampoco podría cubrir la demanda de diferentes colaboradores en gran escala a nivel mundial.

Por lo tanto, se genero una propuesta de mejora para la validación del Proyecto de StarlingX basada en servicios llamada: validación como un servicio (VaaS), por sus siglas en ingles *Validation as a Service*.

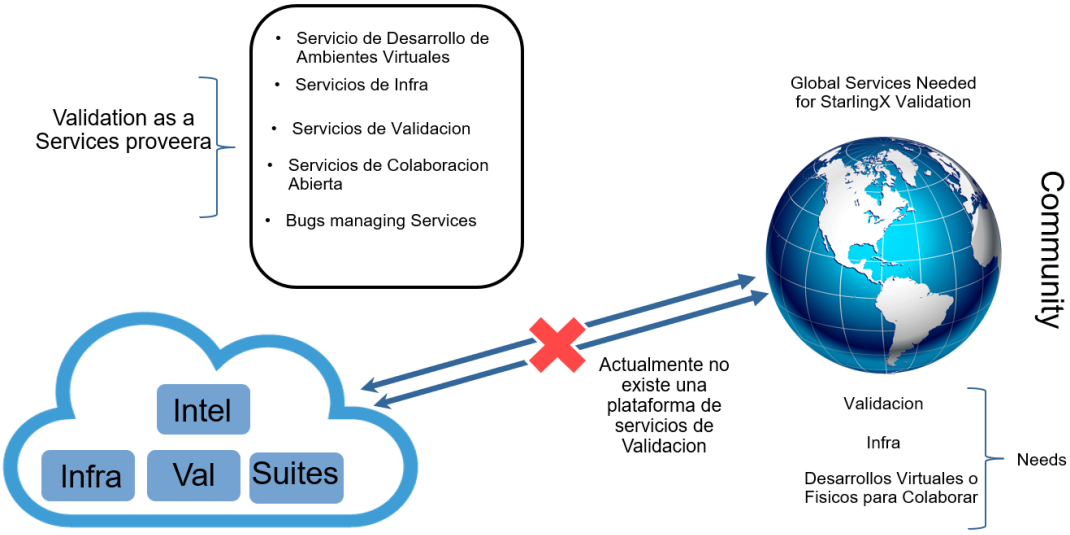


Fig. 1. Sin implementación de propuesta de mejora

En la figura 1 describe que no existe un vinculo o una plataforma de servicios de validación para el proyecto de StarlingX entre Intel su infraestructura y las suites de prueba que puedan trabajar con eficiencia y escalabilidad con todo el mundo. Se pueden observar también las necesidades que tienen por cubrir Intel a nivel mundial como servicios de validación, servicios de infraestructura e implementaciones físicas o virtuales

para poder desarrollar pruebas a los cambios en el código por los desarrolladores a nivel mundial.

Esta propuesta de mejora (VaaS), integraría y proveería: servicios de desarrollo de ambientes virtuales de pruebas, servicios de validación, servicios de colaboración abierta y servicio de administración de errores, todo esto basado en el principio del open source, que es el compartir y la colaboración.

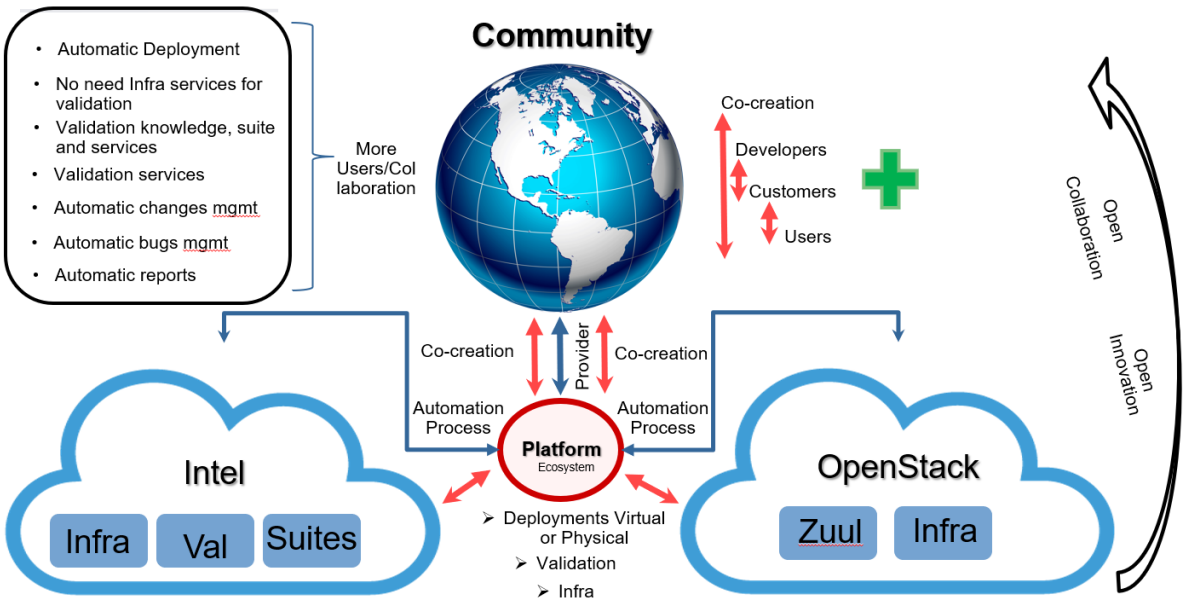


Fig. 2. Implementación con la propuesta de mejora

El principal objetivo de esta propuesta de mejora es un proceso general con diferentes subprocesos automatizados para la integración y entrega continua externa.

Esto quiere decir que será un servicio de validación automatizado para cualquier persona que se encuentre en cualquier parte del mundo que quiera colaborar con este proyecto, en donde todos los servicios de validación serán automatizados sin supervisión, interconectados al mismo tiempo y colaborando con diferentes personas y empresas como Intel y Openstack entre otras.

Las necesidades por cubrir por esta propuesta de mejora es generar una plataforma de servicios la cual pueda brindar:

- ✓ Servicios de Infraestructura virtuales
- ✓ Servicios de validación para la integración Continua
- ✓ Incrementar colaboradores a nivel mundial utilizando la plataforma
- ✓ Dar a conocer tecnologías Edge Computing en el área del Open Source

Algunos de los beneficios que generaría esta propuesta de mejora serian:

- ✓ Reducción de tiempo para el cliente en la implementación de un ambiente de validación
- ✓ Disminución de errores en la integración de código
- ✓ Eliminación de la curva de rampeo en el conocimiento de un Framework de Validación para el nuevo código a integrar para el cliente
- ✓ Validación automática dependiendo las necesidades de los clientes
- ✓ Reporte automatizado en tiempo real solo para el cliente
- ✓ Generar un sistema automático confiable para atraer más colaboradores al proyecto StarlingX

VaaS es un servicio de procesos automatizados el cual proveerá una plataforma de servicios para el proyecto de validación de StarlingX en los cuales los usuarios y sus clientes podrían:

- ✓ Validar sus cambios automáticamente sin tener la Infra
- ✓ Checar si hay algún cambio antes de ser agregado el código de cualquier colaborador
- ✓ Ayudar a hacer debug del proyecto
- ✓ validación de sanidad para cualquier cambio integrado
- ✓ Reporteo de problemas para la comunidad global

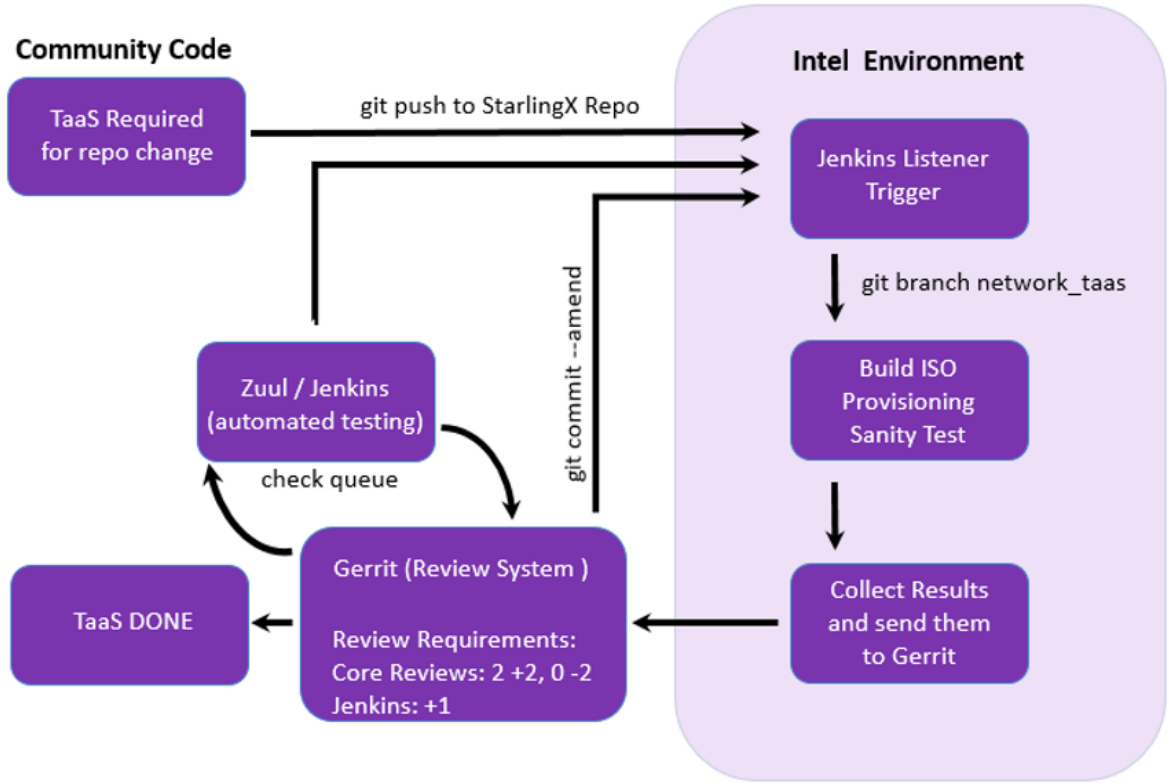


Fig. 3. Primera fase de la propuesta de automatización

Dicha propuesta de mejora no se implementó, debido a motivos de reorganización el proyecto de cambio de sede trasladándose de México a Alemania, por lo cual no de pudo implementar dicha propuesta de mejora sin

embargo en la figura 3, se pueden describir los procesos en bloques de la validación para el código de la comunidad de open source el cual se integraría al proyecto de StarlingX, utilizando diferentes herramientas de integración, tales como: Jenkins, Gerrit, Git y propuestas futuras como pruebas como un servicio (TaaS) por sus siglas en ingles Testing as a Service, el cual se propuso como otro servicio nuevo en un futuro.

4.3 CONCLUSIONES

El proyecto de validacion de StarlingX fue exitoso por si solo desde el momento de desarrollo, implementación hasta su mantenimiento, sin embargo los proyectos siempre están ligados a las necesidades de negocio y organizaciones, desgraciadamente el proyecto de validacion de StarlingX no pudo establecerse dentro de Guadalajara, México por mucho tiempo, el proyecto como tal solo duro 2 años y por necesidades de negocio, Intel decidido mover todos los servicios de validacion, integracion y desarrollo que se hacían en México a Alemania.

Dicho movimiento no permitió la madurez del proyecto ni mucho menos implementaciones de innovación en el proyecto, ya que todas las innovaciones y nuevos procesos, se integrarían al proyecto cuando alcanzara un nivel de madurez, el cual nunca llego por obvios motivos.

Dentro de las empresas, los proyectos y las personas son imprescindibles, por lo tanto, cambios y movimientos de las organizaciones son muy comunes y en lo personal creo que tenemos que estar abiertos al cambio ya que son cosas que nosotros como colaboradores de una empresa no tenemos incidencia ni poder para tomar decisiones en las reorganizaciones.

Aunque el proyecto de validacion de StarlingX se haya transferido a Alemania, dejo un gran conocimiento y buenos momentos entre los

colaboradores, aunque la propuesta de mejora de (VaaS) *Validation as a service* tampoco se implementó, la investigación y el conocimiento que genero esa propuesta fue de mucho valor, tal es el caso que, en conjunto, fueron expuestos en este trabajo como un estudio de caso.

El proyecto al final con lo que se generó en Guadalajara se vio beneficiado al dejar un legado de infraestructura, procesos de CI/CD, procesos de DevOps, framework y suite de pruebas de validación los cuales, en conjunto, mejoraron en forma sustancial en la calidad del código de los repositorios existentes ya en la comunidad *open source* y aparte agregando colaboración, conocimiento y casos de prueba tanto al proyecto de validación y al mismo tiempo a la comunidad Open Source.

Sin embargo, el proyecto de StarlingX sigue activo en Alemania generando colaboración a nivel mundial entre las organizaciones, principalmente en países de Aisa tales como China, Hong Kong e India, utilizado principalmente para sistemas de telefonía y manufactura.

5. BIBLIOGRAFÍA

Bibliography

[1] Smartbear, 2021. [Online]. Available: https://smartbear.com/learn/automated-testing/test-automation-frameworks/.

[2] Techtarget, 2021. [Online]. Available: https://searchdatacenter.techtarget.com/definition/edge-computing.

[3] I. Univertisy, "Telefónica, IE University y Nokia incorporan el 5G y la realidad virtual a la enseñanza," 2025. [Online]. Available: https://www.ie.edu/es/universidad/noticias-eventos/noticias/telefonica-ie-university-nokia-incorporan-5g-realidad-virtual-ensenanza/.

[4] A. Terminals, "Lifting the standard of digital connectivity," 2025. [Online]. Available: https://www.apmterminals.com/en/about/digital-solutions.

[5] Opensource.org, 2020. [Online]. Available: https://opensource.org/history#:~:text=The%20%E2%80%9Copen%20source%E2%80%9D%20label%20was,of%20the%20Netscape%20source%20code..

[6] Stratusmedia, 2022. [Online]. Available: https://www.stratusmedia.io/blog/desarrollo-informatico/que-es-la-tecnologia-open-source/.

[7] opensource.org, "Opensource OSD," 2022. [Online]. Available: https://opensource.org/docs/osd.

[8] T. Informacion, "Validación de Datos," 2018. [Online]. Available: https://www.tecnologias-informacion.com/validacion.html. [Accessed 09 04 2022].

[9] H. I. Industrial, 2020. [Online]. Available: https://www.harbor.com.br/es/harbor-blog/2018/03/15/validacao-de-software/#:~:text=La%20validaci%C3%B3n%20de%20software%20es,seguridad%20y%20trazabilidad%20de%20informaciones..

[10] Software, SmartBear, "https://smartbear.com/," 2020. [Online]. Available: https://smartbear.com/learn/automated-testing/test-automation-frameworks/.

[11] R. H. I. I. A. Equal, 2020. [Online]. Available: https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best#:~:text=The%20SDLC%20process%20involves%20several,%2C%20or%20models%2C%20to%20consider..

[12] Stackify, "Stackify SDLC," [Online]. Available: https://stackify.com/what-is-sdlc/.

[13] SmartBear, 2020. [Online]. Available: https://smartbear.com/learn/automated-testing/software-testing-methodologies/.

[14] P. S. Foundation, 2020. [Online]. Available: https://www.python.org/.

[15] I. Free Software Foundation, 2020. [Online]. Available: https://www.gnu.org/software/bash/.

[16] R. F. Foundation, "robotframework.org," 2020. [Online]. Available: https://robotframework.org/.

[17] Gitlab, 2020. [Online]. Available: https://about.gitlab.com/company/.

[18] AWS, "AWS DevOps," 2022. [Online]. Available: https://aws.amazon.com/devops/what-is-devops/.

[19] Jenkins, 2020. [Online]. Available: https://www.jenkins.io/doc/.

[20] opensource.org, "OpenSource Licenses," 2022. [Online]. Available: https://opensource.org/licenses/.

6. ANEXOS

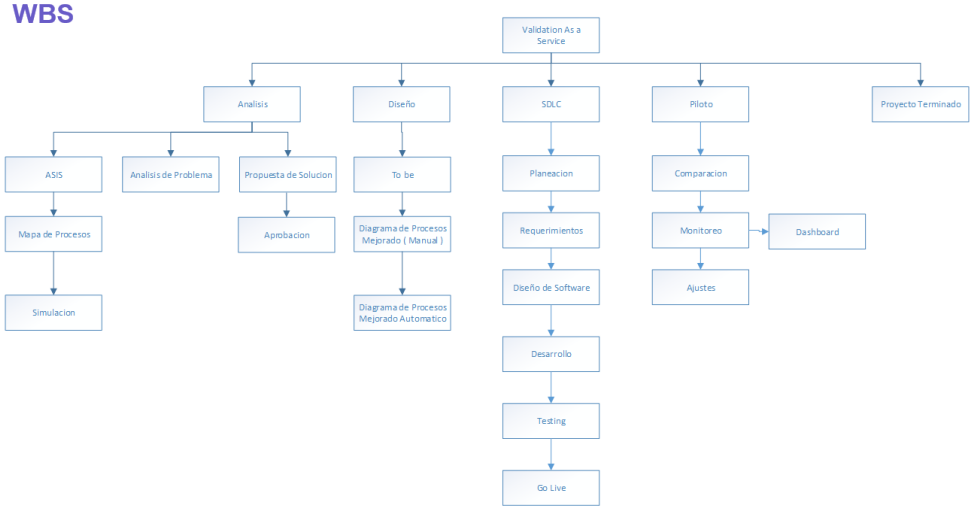
Para el proyecto de validación como servicio, se terminó la etapa de planeación mas no se alcanzó a implementar debido al cambio de proyecto hacia otro país, las siguientes imágenes describen el diagrama a bloques del proyecto Validación como un servicio (VaaS) y el diagrama Gantt

Diagrama a Bloques (WBS)

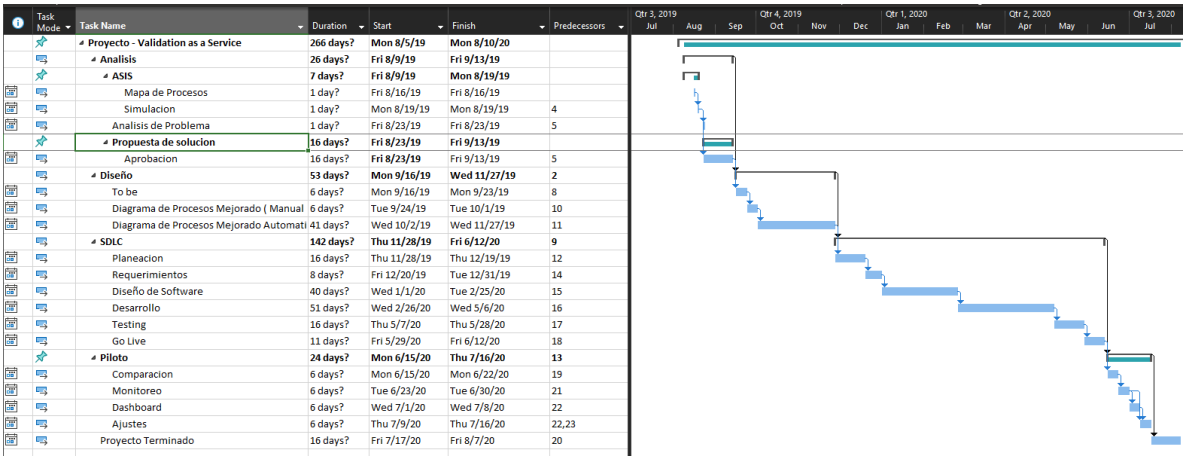
STARLINGX

Validation as a Service Project

WBS



Gantt del Proyecto



Task Mode	Task Name	Duration	Start	Finish	Predecessors
	Proyecto - Validation as a Service	**266 days?**	**Mon 8/5/19**	**Mon 8/10/20**	
	Analisis	**26 days?**	**Fri 8/9/19**	**Fri 9/13/19**	
	ASIS	**7 days?**	**Fri 8/9/19**	**Mon 8/19/19**	
	Mapa de Procesos	1 day?	Fri 8/16/19	Fri 8/16/19	
	Simulacion	1 day?	Mon 8/19/19	Mon 8/19/19	4
	Analisis de Problema	1 day?	Fri 8/23/19	Fri 8/23/19	5
	Propuesta de solucion	**16 days?**	**Fri 8/23/19**	**Fri 9/13/19**	
	Aprobacion	16 days?	Fri 8/23/19	Fri 9/13/19	5
	Diseño	**53 days?**	**Mon 9/16/19**	**Wed 11/27/19**	**2**
	To be	6 days?	Mon 9/16/19	Mon 9/23/19	8
	Diagrama de Procesos Mejorado (Manual	6 days?	Tue 9/24/19	Tue 10/1/19	10
	Diagrama de Procesos Mejorado Automati	41 days?	Wed 10/2/19	Wed 11/27/19	11
	SDLC	**142 days?**	**Thu 11/28/19**	**Fri 6/12/20**	**9**
	Planeacion	16 days?	Thu 11/28/19	Thu 12/19/19	12
	Requerimientos	8 days?	Fri 12/20/19	Tue 12/31/19	14
	Diseño de Software	40 days?	Wed 1/1/20	Tue 2/25/20	15
	Desarrollo	51 days?	Wed 2/26/20	Wed 5/6/20	16
	Testing	16 days?	Thu 5/7/20	Thu 5/28/20	17
	Go Live	11 days?	Fri 5/29/20	Fri 6/12/20	18
	Piloto	**24 days?**	**Mon 6/15/20**	**Thu 7/16/20**	**13**
	Comparacion	6 days?	Mon 6/15/20	Mon 6/22/20	19
	Monitoreo	6 days?	Tue 6/23/20	Tue 6/30/20	21
	Dashboard	6 days?	Wed 7/1/20	Wed 7/8/20	22
	Ajustes	6 days?	Thu 7/9/20	Thu 7/16/20	22,23
	Proyecto Terminado	16 days?	Fri 7/17/20	Fri 8/7/20	20