UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de la herramienta de prevalidación para el equipo de in-vehicle software en Ford de México

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Alonso Rodríguez Castro

ASESOR DE INFORME

M.A. Luis Yair Bautista Blanco

Ciudad Universitaria, Cd. Mx., 2025

Agradecimientos

A mi familia, no habría podido llegar hasta donde estoy si no fuera por ellos; fueron, son y serán la razón por la que he podido lograr todo lo que me he propuesto. Este logro es tanto suyo como mío, no tengo las palabras para expresarles la gratitud que siempre tendré por ayudarme a ser quien soy.

A la Facultad de Ingeniería por haberme dado todo lo que tenían para enseñarme, por haberme permitido conocer a mis amigos, en fin, por darme una experiencia de vida que siempre atesorare.

A mi asesor, por acompañarme y guiarme durante todo este proceso.

Índice

Objetivo

El presente trabajo tiene como objetivo mostrar como los conocimientos adquiridos durante mi formación como Ingeniero en Mecatrónico fueron aplicados para resolver una problemática que existía en el equipo de In-Vehicle Software de la empresa Ford Motor Company México.

Introducción

El programa de la empresa por el cual ingrese a la compañía coloca a cada uno de los participantes en un equipo el cual tiene como meta llevar a cabo un proyecto para implementar mejoras a los procesos internos. El presente reporte describe el desarrollo de la herramienta que se obtuvo para reducir el tiempo de resolución de problemas y mejorar la calidad del trabajo que el equipo entrega.

Perfil de la empresa

Descripción de Ford Motor Company

Ford Motor Company es uno de los principales fabricantes de automóviles del mundo, con una historia de innovación y avance tecnológico que abarca más de 100 años, desde el Modelo T hasta el Mustang, Ford ha producido algunos de los vehículos más icónicos y populares de la historia. Además, la compañía continúa innovando hoy, con un enfoque en el desarrollo de vehículos sostenibles y eficientes.

Henry Ford, un ingeniero y emprendedor de Michigan, estableció la Ford Motor Company en 1903, su objetivo era simple pero ambicioso, fabricar vehículos que fueran asequibles para el ciudadano promedio. A través de la implementación de técnicas innovadoras de producción y la estandarización de los procesos, Ford logró reducir los costos de fabricación y, en última instancia, el precio de venta de sus automóviles, esto fue posible gracias a la famosa filosofía de Ford: "fabricar un automóvil para las masas, no para los ricos".

Historia de la empresa

Junto con 11 socios, Henry Ford, fundó con 28.000 dólares de capital lo que se convertiría en una de las corporaciones más grandes e importantes del mundo. Durante los primeros 15 meses desde su fundación, se vendieron 1.700 unidades del modelo A, siendo el primero de una generación de 19 modelos que seguirían el orden alfabético. Entre los más famosos de esta estirpe estuvo el modelo N que se vendió a 500 dólares

Posteriormente, se produjo un boom comercial con el lanzamiento del Ford T en 1908, marcando una mejora considerable sobre los anteriores modelos. Su éxito se prolongó durante 19 años y las ventas superaron los 15 millones de unidades en todo el mundo. Era definitivo que la creación de esta empresa había generado una verdadera revolución industrial.

En 1917 Ford comenzó a producir camiones y tractores, y dos años más tarde inició la construcción del complejo manufacturero de Rouge en Dearborn, Michigan. Para 1922, la corporación había adquirido la Lincoln Motor Company, y en 1925 construyó el primero de los 196 aeroplanos Ford Trimotor utilizados por las nacientes líneas aéreas comerciales de Norteamérica. Ya en 1927 el modelo T había cumplido su ciclo, lanzándose al mercado el innovador Ford A, vehículo que gracias a sus avances tecnológicos se convirtió en un nuevo éxito pues vendió 4.5 millones de unidades. Para 1932, Ford se convirtió en la primera

compañía en la historia que fundió con éxito un bloque de motor V-8 en una sola pieza, por lo que el modelo Ford V-8 representó el liderazgo automotor durante varios años.

El Mercury lanzado en 1938 fue el último vehículo comercial de esta primera etapa, pues a partir de 1942 Ford Motor Company dedicó todos sus recursos al esfuerzo bélico de Norteamérica, produciendo en menos de tres años 8.600 bombarderos "Liberator" de 4 motores B-24, 57 mil motores para naves aéreas y más de 250 mil jeeps, tanques, destructores de tanques y otras piezas de maquinaria de guerra. [1]

A partir de 1945 Ford volvió a coger vuelo. No fue hasta 1954, con la salida del Ford Thunderbird, cuando decidieron salir a la bolsa. Además, en 1961 también crearon el emblemático Mustang. En 1978 la marca ya tenía más de 40 plantas de fabricación fuera de Estados Unidos y su infraestructura crecía cada vez más. Se sucedían presidentes como Robert McNamara, Lee Laroca o Phillip Ladwell, mientras que a su vez se construyen coches míticos como el Ford Fiesta, Escort o Taurus.

En los siguientes años expandieron el mercado introduciendo vehículos como el Explorer, el Mondeo, la introducción del Lincoln Navigator y el Ford GT en el 2004. Así mismo invirtiendo más en tecnología asociándose con Microsoft para ofrecer SYNC en sus vehículos a partir del 2008 y en el 2009 comienza a ofrecer su línea de motores EcoBoost. Hasta la fecha, Ford continua con su compromiso es ser la compañía más confiable en términos de movilidad y diseño de vehículos inteligentes que ayuden a las personas a transportarse de manera libre y segura. [2]

Ford Motor Company en México

El 23 de junio del 2020, Ford cumplió 95 años en México, fecha en la que la firma del óvalo azul llegó a transformar la movilidad como la primera empresa automotriz en establecerse en el país, convirtiéndose así en precursor de la creciente industria automotriz.

Un 26 de agosto, pero de 1925, inició la historia y relación de Ford con México con la inauguración de la primera planta de ensamblaje de Ford en el país, encargada del montaje y acabado de automóviles en calzada de Balbuena y prolongación Candelaria en la Ciudad de México. Esta planta inició produciendo cinco Ford Modelo T al día, modelo que enseñó el arte de conducir a numerosos mexicanos y estandarizó el volante a la izquierda, diseño que fue imitado posteriormente por la totalidad de los fabricantes, con excepción de los ingleses.

La decisión de Henry Ford de abrir una pequeña planta ensambladora en la ciudad de México, resultó fundamental. Por un lado, esa inversión significó el inicio de la industria automotriz mexicana y, por otro, fue el detonador de nuevas y mayores inversiones industriales y financieras. Con el paso del tiempo, la producción de Ford se trasladó a las afueras de la ciudad y con ello, paulatinamente, se fue incrementando el número de unidades que se producían de las plantas mexicanas hacia el resto del mundo, desde el Modelo A, Cougar, hasta Mustang.

El número de automóviles y camiones Ford que circulaban por esas fechas en el país, había desbordado la capacidad de los importadores y concesionarios para atender las necesidades de mantenimiento, refacciones y control de calidad de todos esos vehículos. Por tal razón, uno de los retos que enfrentó Ford para establecer su primera planta en México, fue la incorporación de mano de obra capacitada y la proveeduría de insumos y piezas básicas para el funcionamiento de los automóviles. Para resolver el primero, se decidió repatriar a obreros y técnicos mexicanos que trabajaban como inmigrantes en las plantas de Detroit, mientras que para el segundo se hizo trabajo previo de desarrollo de proveedores de neumáticos y otros aditamentos.

La planta ensambladora de San Lázaro, en la que laboraron 260 trabajadores, inició con la producción de cinco vehículos Modelo T por día. Para esa época, Ford preparó para su circulación 3,200 autos y 1,943 camiones. Los simpáticos modelos T en sus múltiples variantes, como el sobrio Tudor sedán, cuyo costo era de 1,894 pesos y descapotable Touring de 1,035 pesos, formaba, junto con taxis y camiones de la misma plataforma, parte del paisaje citadino.

En la actualidad, llegando a la segunda mitad del año 2020, Ford cuenta con cuatro plantas en distintos puntos del país, con un aproximado de 10,197 empleados en México, un centro de ingeniería que cuenta con alrededor de 2,000 ingenieros que desarrollan productos y patentes para el país y para el mundo, con una clara visión hacia el futuro. [3]

Desde su llegada a México, hace 97 años, Ford Motor Company ha tenido como objetivo lograr un impacto positivo en la comunidad mexicana, fomentando la educación, ofreciendo mejores oportunidades de empleo y haciendo énfasis en el cuidado al medio ambiente. Bajo este enfoque es que se creó el Centro Global de Tecnología y Negocios (GTBC, por sus siglas en inglés), el nuevo Campus de Ford que se compone de oficinas corporativas y uno

de los mayores centros de ingeniería en Latinoamérica ubicado en Naucalpan de Juárez en el Estado de México.

Actualmente el GTBC de Ford en México se involucra en 8 de cada 10 programas internacionales de Ford Motor Company; lo que lo convierte en una pieza clave en el desarrollo de Tecnología. En ese sentido, el talento y trabajo de los mexicanos es de suma importancia, pues están involucrados hasta 600 ingenieros en cada proyecto. Desde SUVs y Crossovers hasta la primera pickup eléctrica, los ingenieros mexicanos participan activamente en el desarrollo de Vehículos de última generación; ejecutando hasta el 70% de la ingeniería de Vehículos como Ford Bronco Sport, Ford Expedition y Lincoln Navigator. Además, contribuyen al desarrollo de Eléctricos como Ford F-150 Lightning y Ford Mustang Mach-E.

Desde 2016 ha generado más de 500 patentes registradas ante el Instituto Mexicano de la Propiedad Industrial (IMPI) y desarrollado más de 4,700 ideas en proceso de registro. Además, ha desarrollado más de 490 mejoras a los procesos globales de la compañía. [4]

Figura 1. Vista aérea del GTBC México [5]

Descripción del Puesto

Conocimientos generales como Ingeniero Mecatrónico

Mi formación como Ingeniero Mecatrónico me ha proporcionado una base multidisciplinaria y versátil, esencial para abordar los desafíos complejos que presenta la industria automotriz. Esta formación, combinada con mi enfoque en la programación, ha sido fundamental para mi desempeño en Ford Motor Company. Mis conocimientos abarcan:

- Diseño y manufactura: Entendimiento de los procesos de diseño, modelado y simulación de componentes y sistemas mecánicos, eléctricos y electrónicos. Familiaridad con software CAD (SolidWorks, AutoCAD) y herramientas de simulación para analizar sistemas dinámicos y térmicos. Conocimiento de procesos de manufactura convencionales y no convencionales (maquinado CNC, impresión 3D, soldadura robótica).
- Control y automatización: Habilidad para diseñar, implementar y mantener sistemas de control automático, utilizando herramientas de programación y control de procesos. Diseño de controladores PID, implementación de sistemas de control basados en lógica difusa y redes neuronales, y configuración de sistemas de control distribuido (DCS) y controladores lógicos programables (PLC).
- Programación: Dominio de lenguajes de programación de alto nivel, incluyendo Python (utilizado extensivamente en el desarrollo de la herramienta de verificación IVS), C++, y conocimientos básicos de Java. Familiaridad con los paradigmas de programación orientada a objetos (POO) y programación funcional.
- Estructuras de datos y algoritmos: Conocimiento profundo de estructuras de datos fundamentales como listas, diccionarios, árboles y grafos, así como algoritmos de búsqueda, ordenamiento y optimización.
- Bases de datos: Familiaridad con sistemas de gestión de bases de datos (DBMS) relacionales (MySQL, PostgreSQL) y no relacionales (MongoDB). Conocimiento del lenguaje SQL para la consulta y manipulación de datos, así como los principios de diseño de bases de datos.
- Automatización web (Web Scraping): Experiencia en el uso de librerías como Selenium para la automatización de navegadores web y la extracción de datos de sitios web.
- Desarrollo de interfaces gráficas (GUI): Experiencia en el desarrollo de interfaces gráficas de usuario utilizando librerías como Tkinter y Qt.

- Control de versiones (Git): Dominio de herramientas de control de versiones como Git para la gestión del código fuente y la colaboración en equipos de desarrollo.
- Testing y depuración: Experiencia en la creación de pruebas unitarias y pruebas de integración para garantizar la calidad del código. Conocimiento de herramientas de depuración para identificar y corregir errores en el software.
- Metodologías de desarrollo de software: Familiaridad con metodologías ágiles de desarrollo de software como Scrum y Kanban.
- Conocimientos específicos de la industria automotriz: Entendimiento de los protocolos de comunicación utilizados en los vehículos (como CAN bus), así como los estándares de seguridad y calidad aplicables al desarrollo de software automotriz.
- Análisis y solución de problemas: Capacidad para identificar, analizar y resolver problemas complejos en sistemas mecatrónicos y de software, utilizando metodologías de ingeniería y herramientas de simulación.
- Gestión de proyectos: Conocimientos básicos sobre la planificación, ejecución y control de proyectos de ingeniería y software, incluyendo la gestión de recursos y la coordinación de equipos de trabajo.

Estos conocimientos, combinados con mi capacidad de aprendizaje continuo, mi creatividad para resolver problemas, mi pasión por la programación y mi compromiso con la innovación, me han permitido desarrollar soluciones de software innovadoras y eficientes para Ford Motor Company, contribuyendo a la mejora de la calidad y la eficiencia en los procesos de liberación de software.

Actividades específicas de las posiciones

Durante el tiempo que llevo trabajando en la empresa he ocupado dos diferentes posiciones, las cuales se relacionan entre ellas, debido a que, a partir de los resultados entregados en la primera posición fue como pude obtener la segunda.

Ingeniero de Software en el Programa “Product Development (PD) Academy” Ford México

Mi primera etapa en Ford consistió en un programa de reciente creación, enfocado en recién egresados o profesionales con un máximo de dos años de experiencia. Los participantes son asignados a equipos dentro de las áreas de software o hardware, según las necesidades.

El programa cuenta con la participación de un comité conformado por supervisores y gerentes de diferentes áreas de ingeniería dentro de Desarrollo de Producto, quienes tienen la flexibilidad de dividirse los roles esenciales. Además, el equipo de Atracción de talento maneja todo el proceso de reclutamiento y selección de los candidatos, mientras que un equipo de Recursos Humanos ofrece soporte durante todo el programa, asegurando que los participantes tengan una experiencia integral y enriquecedora. Con duración de 6 meses, este sistema consiste en una incubadora de talento en donde a cada participante se nos asignó un proyecto específico para desarrollar dentro de la organización, este proyecto tiene el fin de obtener como resultado una innovación que impacte ya sea a los productos o los procesos de Ford de México. [6]

Ingeniero de Software en el Vehículo (Ingeniero de IVS, por sus siglas en inglés)

En mi posición actual dentro del equipo de IVS (In-Vehicle Software), trabajo en un entorno donde nos dedicamos al correcto funcionamiento del software de los vehículos. Este equipo tiene las siguientes responsabilidades:

- **Liberación de software para vehículos en producción**. Preparación, validación y entrega del software a los vehículos que se están fabricando, asegurando el cumplimiento de los estándares de calidad y funcionalidad.
- **Actualizaciones de software para vehículos ya vendidos o en venta.** Desarrollo, prueba y distribución de actualizaciones para mejorar el rendimiento, incorporar nuevas funciones y corregir posibles errores.
- **Atención a las alarmas generadas al final de la línea de producción (End-Of-Line, EOL):** Investigación y solución de errores o anomalías detectadas en el software durante el proceso de producción, garantizando la calidad del producto final.
- **Ofrecer actualizaciones al software de los vehículos ya vendidos.** Desarrollo, prueba y distribución de actualizaciones para mantener los vehículos actualizados con las últimas tecnologías y mejoras.
- Mantenimiento de un registro de las versiones de software que cada ingeniero de diseño (D&R) desea implementar para su respectiva Unidad de Control Electrónica (ECU): Seguimiento de las diferentes versiones del software utilizadas en cada módulo del vehículo, garantizando la compatibilidad y evitando problemas de integración.

- Coordinación para asegurar que las versiones o números de parte de cada software instalado en los módulos de los vehículos en producción sean consistentes en todas las bases de datos: Mantener la integridad de la información y evitar confusiones durante el proceso de producción y la gestión de los vehículos.

Gracias a los resultados obtenidos en mi posición anterior, mis tareas dentro de este equipo se centran en la automatización de procesos y tareas manuales, con el objetivo de mejorar la calidad del trabajo, reducir errores humanos y disminuir el tiempo de ejecución. Además de las tareas generales del equipo, me dedico al desarrollo de aplicaciones o herramientas de software para cumplir con las asignaciones encomendadas al aceptar esta nueva posición. Para alcanzar mis objetivos como ingeniero de software, llevo a cabo las siguientes actividades específicas:

- Adquirir el mayor conocimiento posible del proceso que se pretende mejorar o automatizar para tener una perspectiva como usuario potencial, entendiendo que para crear una solución efectiva, es necesario comprender a fondo el problema desde la perspectiva del usuario final.
- Ofrecer diversas soluciones para la problemática identificada con el objetivo de explorar diferentes alternativas y proponer soluciones innovadoras que aborden la problemática de forma eficiente.
- Diseñar un plan con las tareas a realizar, estableciendo objetivos a cumplir dentro de un período de tiempo específico. La planificación es fundamental para garantizar que el proyecto se desarrolle de forma organizada y eficiente, alcanzando las metas establecidas en el tiempo previsto.
- Seleccionar las herramientas, o en este caso, los lenguajes de programación, que se utilizarán para desarrollar las soluciones propuestas. La elección de las herramientas adecuadas es esencial para el éxito del proyecto.
- Coordinar con otros ingenieros para la distribución e integración de las tareas necesarias para el cumplimiento del proyecto. El trabajo en equipo es fundamental para el éxito de cualquier proyecto. Mi rol implica colaborar estrechamente con otros ingenieros para garantizar que las tareas se completen de forma eficiente y que las soluciones se integren correctamente.

Proyecto desarrollado

Herramienta de Verificación del equipo IVS (IVS Verification Tool)

Contexto

Durante mi primera etapa en la empresa, a través del programa Ford PD Academy, cada persona tenía asignado un proyecto específico para llevar a cabo dentro del equipo al cual era asignado. En mi caso, este proyecto fue el desarrollo de la herramienta de software que es el tema central de este documento.

Al ser una persona ajena a la organización y sin conocimientos del área que utilizaría el resultado del proyecto ni de los términos utilizados dentro del equipo, mi abordaje inicial consistió en conocer lo mejor posible el problema. Para esto, me planteé un conjunto de preguntas que me servirían como punto de partida para el diseño de la solución de software:

- ¿Cuándo se presenta dicho problema?
- ¿Por qué se presenta?
- ¿Cuál es la manera más eficiente de resolverlo?
- Si la herramienta funciona como una caja negra:
    - ¿Qué es lo que se requiere como entrada?
    - ¿Qué es lo que se esperaría como salida?
    - ¿Cuál sería el formato de dicha salida?
    - ¿Qué información debe contener dicha salida?

Para responder estas preguntas, solicité a mi superior que se me asignaran las mismas tareas que realizan los miembros del equipo. Mi solicitud fue aprobada y comencé el mismo entrenamiento que reciben los nuevos integrantes del equipo, el cual duró alrededor de 3 meses. Al finalizar este período, era capaz de realizar las mismas tareas que mis compañeros. A partir de este entrenamiento, obtuve las respuestas a las preguntas iniciales y comprendí quiénes serían los usuarios del proyecto, lo que me permitió identificar otros requerimientos no contemplados inicialmente:

1. El formato de salida requerido variaba entre los usuarios, por lo que se acordó un formato estándar funcional para la mayoría de los casos, exceptuando situaciones especiales.

2. La herramienta inicialmente analizaría un solo elemento a la vez (un modelo de vehículo con un solo año), por lo que los usuarios solicitaron la posibilidad de analizar múltiples elementos a la vez (un modelo de vehículo con múltiples años).
3. Con el punto anterior, el formato de salida cambió, dando lugar a una organización de archivos de salida diferente.
4. Se propuso que la herramienta descargara el reporte automáticamente, ingresando el modelo y año(s) por analizar.

La duración del programa Ford fue insuficiente para terminar este proyecto debido a su alcance. Sin embargo, gracias a la importancia y los resultados que estaba entregando, la empresa me ofreció una vacante para continuar desarrollando esta aplicación, así como otras ideas que se tenían. En resumen, este proyecto fue mi puerta de entrada a la empresa y el medio por el cual obtuve una posición fija, siendo el proyecto más amplio y relevante que he desarrollado hasta ahora.

Introducción al Problema

El sistema eléctrico de un vehículo se gestiona y controla a través de Unidades de Control Electrónico (ECU). Una ECU es un dispositivo integrado que controla uno o más sistemas o subsistemas eléctricos de un vehículo. Por ejemplo, en Ford, el PSCM (Módulo de control de dirección asistida) es la ECU responsable del sistema de dirección, mientras que el TCM (Módulo de control de transmisión) gestiona la transmisión del vehículo. Ford tiene más de 90 ECU diferentes, cada una con su correspondiente Paquete de Software (Paquete SW), que se actualiza constantemente a medida que la tecnología de los vehículos evoluciona. Estas actualizaciones pueden ser simples correcciones de errores o actualizaciones de software completas de múltiples módulos.

Como se mencionó anteriormente, el equipo de IVS se encarga de garantizar que toda la información de una versión de software específica esté completa y lista para su uso en vehículos de producción o en circulación.

Planteamiento del Problema

Cuando se liberan nuevas versiones de software, el cambio se realiza fácilmente si el producto aún se encuentra en producción, ya que la empresa posee el vehículo y puede instalar el nuevo software desde la base de datos de la planta. Sin embargo, el proceso para actualizar el software de los vehículos que ya están en circulación o en venta (concesionarios) es más complejo.

Este proceso se conoce como **Acción de Servicio (Service Action)** y se solicita a través de una Plantilla de Documento de Servicio en la que ingeniería da instrucciones explícitas en una Matriz de Reemplazo (matriz R&R) para reemplazar uno o más ensambles de ECU por uno nuevo, donde el cambio es simplemente una cuestión de reprogramación del módulo.

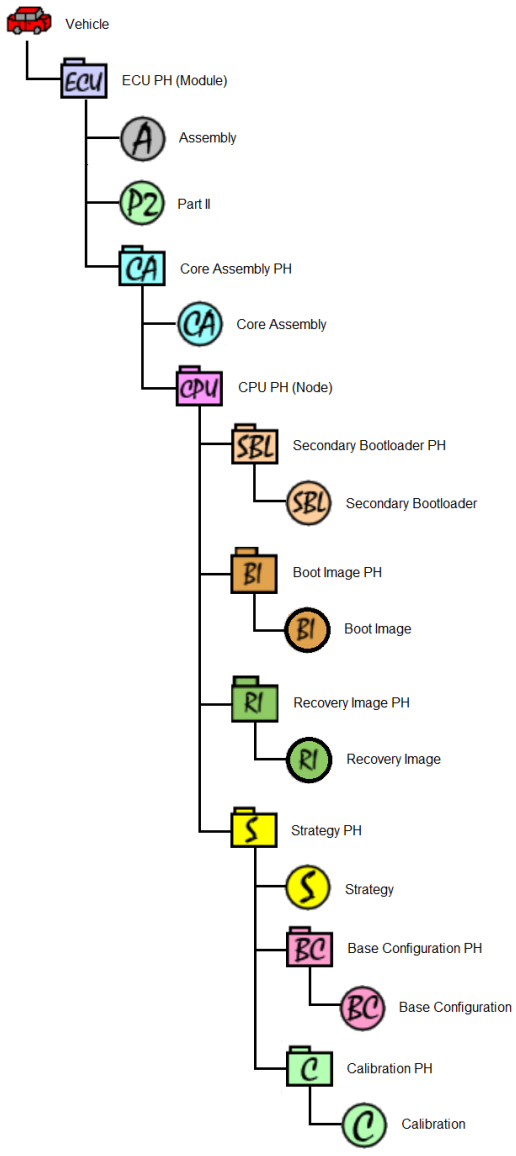

Figura 2. Estructura básica de un ensamble de software

La referencia para este proceso es el **Ensamble del ECU (ECU Assembly)**, un "contenedor" que incluye una combinación específica de elementos que representan el paquete de software de una ECU. Los elementos de los paquetes de software están

representados por números de pieza, y cada número de pieza tiene un código de color y un código de nombre de archivo en la plataforma especializada de IVS.

Cuando se encuentra un problema en el campo de una ECU programable, es posible que se pueda resolver mediante una actualización de software. Incluso si se requiere el reemplazo de un módulo, es posible que solo requiera una instalación de configuración. En estos casos, se utiliza una Acción de Servicio. Hay dos tipos principales de procedimientos de servicio que se realizan en módulos programables:

- Instalación de módulos programables (PMI): Proceso para reemplazar físicamente un módulo defectuoso en el compartimento de servicio y requiere programación. El módulo puede variar desde ser parcialmente funcional hasta ser completamente inoperable.

  Hay dos consideraciones clave para el PMI:
  - Se debe reemplazar el hardware del módulo: el objetivo de la reparación tendrá un hardware diferente al que se instaló cuando se produjo el vehículo.
  - El objetivo final de reparación depende de la política local del FCSD (Ford Customer Service Division).
- Reprogramación de Módulos (MR): Incluye aquellas reparaciones en las que Ingeniería proporcionó información para reprogramar el módulo con un nuevo paquete de software (representado por un nuevo ensamblaje)

Se pretende que toda la información necesaria para que el técnico realice cualquiera de estas reparaciones se origine con la descarga de datos de IVS al servicio para permitir que las herramientas del servicio actualicen el software sin ninguna otra intervención o corrección humana.

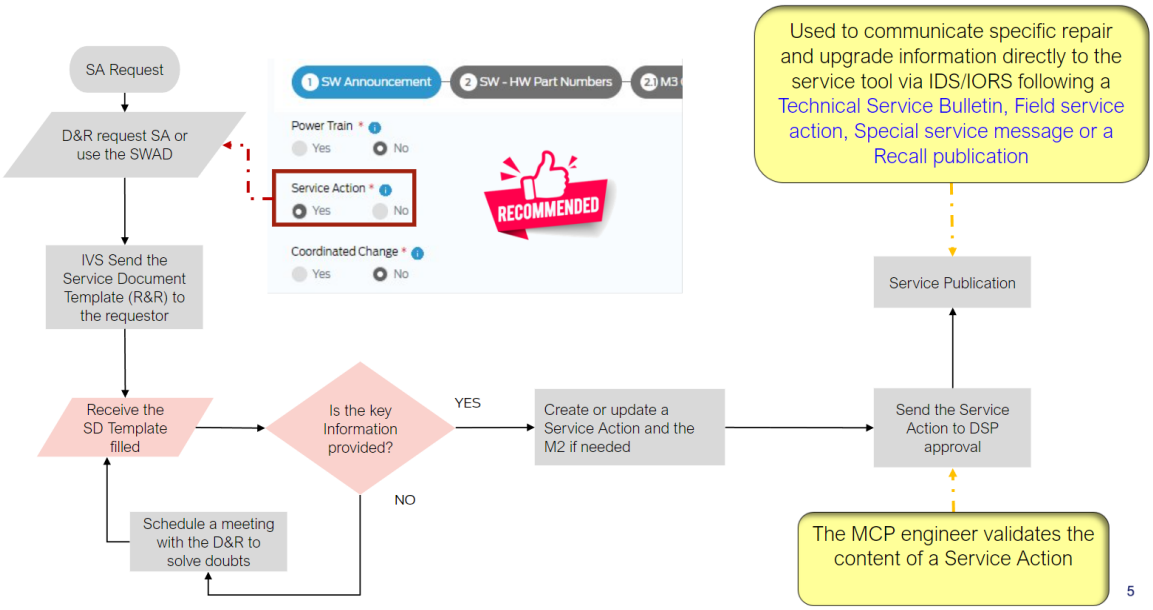

Figura 3. Ciclo de vida de una Acción de Servicio

El problema era que, al terminar el proceso de liberación de las partes del software solicitado por ingeniería, la Acción de Servicio era enviada al equipo de Productos de Servicio de Diagnóstico (DSP). Sin embargo, en la mayoría de los casos, la validación realizada por este equipo fallaba frecuentemente, es decir, los vehículos que debían recibir la actualización no la podían obtener debido a inconsistencias en la estructura del paquete de software. El proceso de validación de una Acción de Servicio es laborioso y puede retrasar la liberación del software.

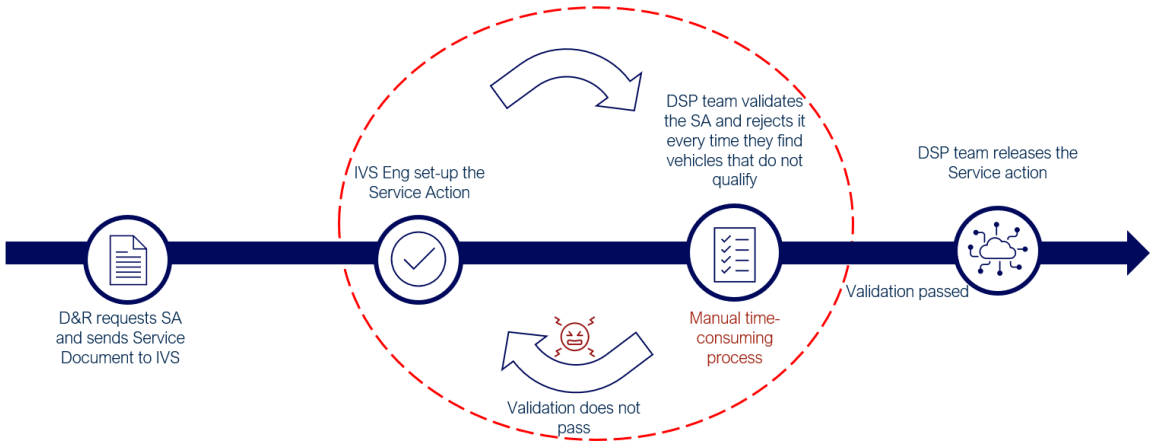

Figura 4. Proceso de validación de las Acciones de Servicio previo

El equipo de IVS no realizaba ningún proceso de prevalidación, es decir, realizaba las modificaciones necesarias en la base de datos según la matriz R&R y enviaba la Acción de

Servicio al siguiente paso del proceso. Cuando el equipo de DSP realizaba la validación de los vehículos por actualizar, se encontraban con que varios de estos presentaban inconsistencias en la estructura del software, por ejemplo, cuando el ensamble por reemplazar no se encuentra en la matriz R&R, o el ensamble está incompleto dentro de la base de datos.

Existen varias razones por las cuales un vehículo presenta estas alertas o alarmas, que impiden que sea elegible para la Acción de Servicio y, por ende, que esta se tenga que regresar a un paso anterior.

El proceso para buscar las causas raíz de estas alertas era laborioso, manual y repetitivo, razones por las cuales no se realizaba dicho procedimiento. El equipo se enfocaba en hacer que las Acciones de Servicio avanzaran a la siguiente etapa sin considerar el porcentaje de vehículos que pasarían las validaciones siguientes.

En resumen, cuando llegaba una Acción de Servicio al equipo de IVS, se realizaban los pasos requeridos para mandarla a la siguiente etapa, lo que frecuentemente ocasionaba que esta regresara debido a alarmas descubiertas en el proceso de validación realizado por el equipo de DSP. El equipo de IVS era capaz de realizar un proceso de prevalidación para evitar que llegaran las alarmas al siguiente equipo, pero este proceso requería mucho trabajo, extendiéndose incluso a semanas, lo que consumía tiempo y retrasaba las liberaciones de software, que en algunos casos podían ser urgentes.

El proceso de prevalidación consiste en los siguientes pasos:

1. Descarga de un reporte que contiene una lista de vines representativos para todos los vehículos de un modelo y año específico.
   Dicho reporte puede contener una lista desde 3 hasta 400 vines representativos, como en el caso de los modelos más vendidos de la empresa, como la camioneta F-150. Este reporte contiene la estructura de software que posee cada vin representativo, es decir, por cada fila se encuentra asociado a cada VIN la cantidad de vehículos representados y la estructura de software que este posee.

Total Vehicles	Total Assemblies	Vehicle Sample	Strategy	Image	Hardware	Core Assembly	Assembly	Calibration	ECU Configuration

Figura 5. Columnas iniciales contenidas en el reporte por analizar

2. Extracción de la lista de vines representativos.

3. Descarga, por cada vin representativo, de un archivo XML con todas las características de software que el vehículo posee.

   La descarga de estos archivos es una de las partes más laboriosas del proceso, debido a que la cantidad de archivos es variable y depende del número de vines obtenido en el reporte inicial. Se tenía que ingresar manualmente cada vin representativo en la base de datos y dar clic en el botón de descarga. Al analizar dos modelos de carro, con más de 600 vines representativos por modelo, este proceso no se realizaba de manera adecuada por el cansancio y la fatiga generada por un proceso tan repetitivo y manual.
4. Con los archivos XML descargados, se tiene que ingresar de manera individual a cada archivo en busca de la existencia de un campo de alarmas (Warnings). Al llevar a cabo el análisis para cada vin representativo se tienen 3 posibles casos:
   a) Si el vehículo presenta el campo de Warnings, se marca en el reporte como “Vehículo con alarma”. Lo que llevaría a una investigación de la razón de dicha alarma y a su respectiva solución.
   b) Si el vehículo no presenta campo de Warnings y no es elegible para la Acción de Servicio. Este caso corresponde a una situación neutral en donde el vehículo no presenta ninguna alarma y además no es elegible para la Acción de Servicio, esto último puede deberse a que el ingeniero que inició el proceso decidió no incluir dicho vehículo en la Acción de Servicio o que simplemente olvidó añadirlo en el documento R&R; esta situación se soluciona confirmando si la información entregada en el R&R es correcta, es decir, el ingeniero está de acuerdo en que el vehículo no sea elegible, o si requiere alguna corrección para añadir el vehículo a la Acción de Servicio.
   c) Si el vehículo no presenta el campo de Warnings y si es elegible para la Acción de Servicio, situación en donde solamente se verifica que todo esté en orden respecto a la estructura de software del vehículo.

Este proceso se realizaba cada vez que una Acción de Servicio era devuelta del equipo de DSP, lo que se traducía en un retrabajo por parte del equipo de IVS para eliminar las alarmas. Sin embargo, el proceso de análisis era laborioso, por lo que el ciclo de enviar la Acción de Servicio, realizar su validación y regresarla debido a las alertas, podía repetirse más de una vez hasta que todos los vehículos elegibles estuvieran limpios de alertas.

Solución

La compañía tiene un enfoque en reducir el tiempo de liberación de software, la cantidad de problemas registrados debido a las alertas y varias otras métricas enfocadas hacia una empresa esbelta (Lean). Por lo tanto, la solución planteada fue la automatización del proceso de prevalidación mediante herramientas de programación para desarrollar una aplicación que pudiera realizar el análisis de forma automática, indicando solamente el modelo del vehículo, el año y la ECU a la que se le realizará la Acción de Servicio.

El proceso de automatización consistió inicialmente en entender el proceso mismo y sus características, con el objetivo de comprender las necesidades de la automatización y la mejor forma de abordarlas.

Después de la etapa de aprendizaje, empecé a crear el algoritmo para completar la tarea de automatización, basado en el proceso, pero contemplando casos especiales que podrían surgir, como módulos o ECUs con características especiales en la estructura del software o en el archivo XML correspondiente. En esta parte, encontré dificultades al automatizar todos los módulos debido a mi poca experiencia para distinguir ciertos aspectos que una persona experimentada está acostumbrada a encontrar.

La aplicación se desarrolló utilizando Python, por las siguientes razones:

- Python cuenta con librerías que facilitan la manipulación de diferentes tipos de archivos (Excel y XML), y al ser un lenguaje de alto nivel, permite un manejo de archivos entendible sin comandos complejos.
- Para la interfaz de usuario, se eligió la librería de Python para interfaces gráficas (Tkinter) en lugar de un lenguaje de programación enfocado a esta tarea (HTML o C#), para evitar una etapa de integración e intercambio de información entre lenguajes que podría generar problemas de comunicación o pérdida de información.
  .
- Como gran parte del proceso requería extraer información y archivos de bases de datos que no son Relacionales o No Relacionales, sino sitios web internos de acceso limitado, se utilizó la librería Selenium para la navegación en sitios web y la extracción de archivos necesarios.
- Se consideró el escenario en donde yo ya no estuviera en el equipo o en la compañía, por lo que Python, al ser un lenguaje accesible, permite que el

mantenimiento y las mejoras de la aplicación puedan ser realizados por otra persona con conocimientos básicos del proceso y del lenguaje de programación.

Otra ventaja que se encontró de fue el hecho que este lenguaje de programación era capaz de ser portable, esto se resolvió con la librería de PyInstaller, la cual permite generar archivos del tipo ejecutables (.exe) para que el código pueda ser compartido hacia cualquier persona sin la necesidad de que este usuario tuviera instalado el lenguaje de Python en su dispositivo.

Al tener contacto directo con los usuarios, era necesario implementar una interfaz gráfica (GUI). Se optó por la librería Tkinter de Python, que permite crear GUI básicas y lo suficientemente complejas para realizar los comportamientos planeados. Esta decisión permitió que las variables obtenidas mediante la interfaz gráfica tuvieran comunicación directa con el backend que se estaba creando. Algunas de las funciones que esta librería ayudó a implementar fueron:

- Crear entradas para ingresar los campos de usuario y contraseña.
- Seleccionar carpetas, es decir, obtener la ubicación de una carpeta.
- Desplazamiento entre las diferentes ventanas o etapas de la aplicación.

Después de decidir el entorno de desarrollo y las herramientas, se diseñó un flujo de trabajo que seguiría la aplicación, considerando los posibles casos que pudieran surgir durante la ejecución de la herramienta.

Los escenarios representados son los que dan la pauta para cada una de las ventanas que se necesitan crear en la interfaz. Cada caso contempla entradas y decisiones diferentes, es decir, conforme el programa avanza, se piden los valores de entrada y se guardan para que al ejecutar el análisis, la aplicación sepa cuál de los siguientes casos es el que se llevará a cabo. Se tuvieron, en general, cuatro casos posibles:

1. No se tiene reporte descargado y no se tienen archivos XML descargados: En este caso, no se tiene ningún archivo, por lo que se requiere descargarlos para realizar el análisis. Se preguntan los datos necesarios para la descarga del reporte y, posteriormente, las características que se desea que cada archivo XML contenga.
2. No se tiene el reotre descargado pero si los archivos XML: Este caso es el menos probable, ya que los archivos XML no se suelen tener descargados con anticipación. Sin embargo, se decidió considerarlo porque se dieron algunos casos en donde se

cambiaba un archivo XML por alguno actualizado y los demás permanecían igual. Para no descargar todos los archivos, este escenario fue la manera de resolverlo.

3. Se tiene el reporte descargado pero no los archivos XML: En esta secuencia se le pide al usuario que indique cual es el archivo que desea utilizar, despues, indicar las caracteristicas de los archivos XML que se van a utilizar.
4. Se tiene el reporte y los archivos XML descargados: Para este escenario, al tener todos los elementos necesarios para el analisis, solamente se extraia la información necesaria del reporte y se procede al analisis directamente.

Para realizar el proceso lo más automatizado posible, todas las entradas mencionadas se recababan al inicio de la aplicación y, al oprimir el botón de ejecutar proceso, se hacía la comparación de las respuestas obtenidas por el usuario para saber qué escenario debía correr la aplicación. Cada respuesta generaría una ventana diferente para ingresar los datos requeridos en ese caso seleccionado, así que se creaba una ventana para cada uno de los pasos del diagrama de flujo.

Aun al considerar los diferentes casos que se tienen en cuanto a la forma en la que las entradas del sistema son obtenidas, un punto ineludible para todas las situaciones es el análisis que se realiza, ya que este es el proceso que se desea de la aplicación.

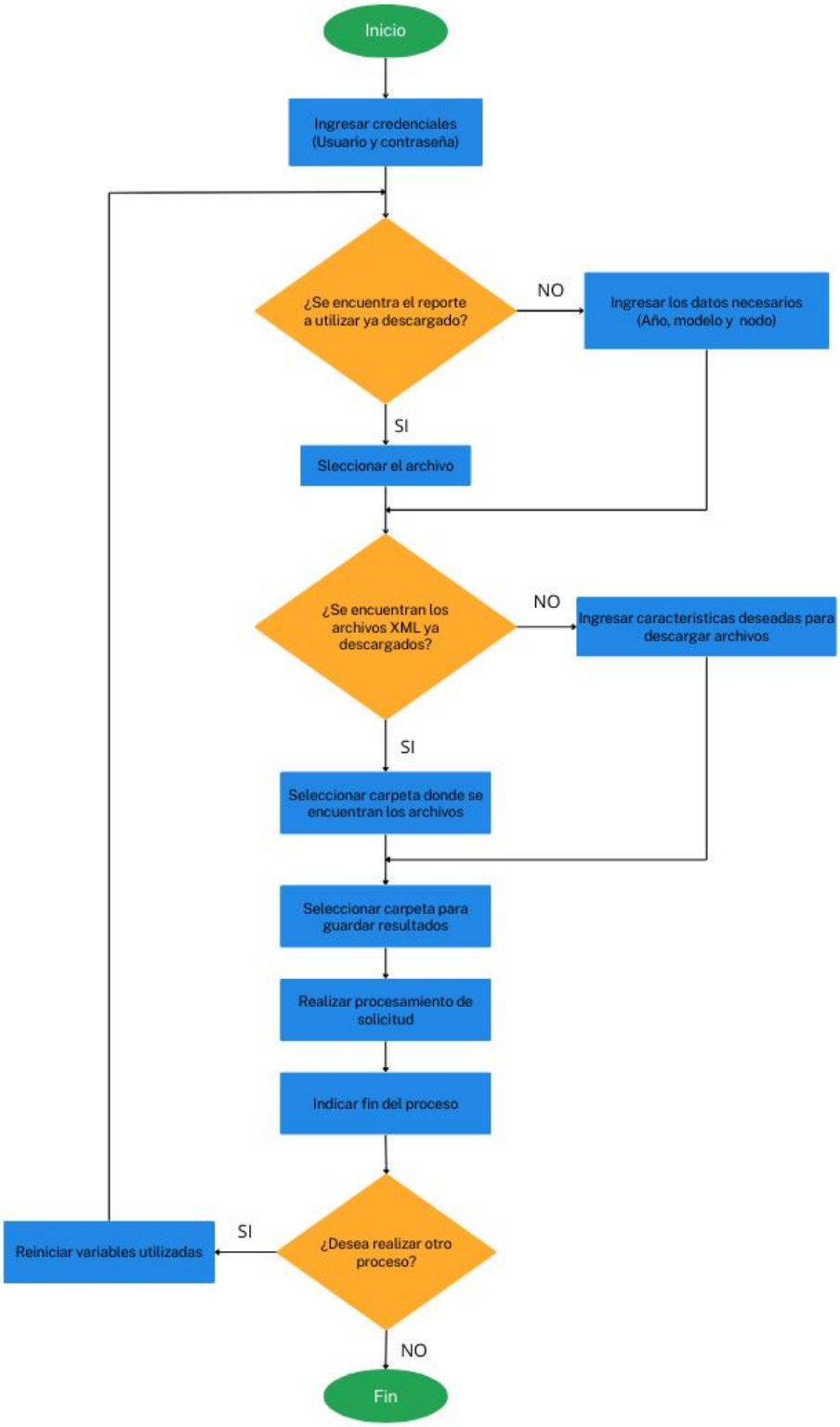

Diagrama 1. Flujo seguido por la aplicación

A partir de lo mencionado, las entradas que se le solicitaran al usuario en cada paso son las siguientes:

1. Solicitud de las credenciales necesarias para ingresar a las bases de datos de las cuales se descargará los archivos necesarios para el análisis.

Figura 6. Ventana de la aplicación para ingresar las credenciales

2. Se preguntará si el reporte con la lista de vines ya se encuentra descargado o si se desea analizar.
    a. Para el caso en el que el reporte ya se encuentre descargado, se le preguntara al usuario que seleccione de sus archivos la ubicación de este.
    b. Para el segundo caso en donde no se tiene el reporte descargado, se pedirá al usuario ingresar el modelo, año y modulo que se desean analizar (se puede realizar el análisis de múltiples años en caso de ser necesario).

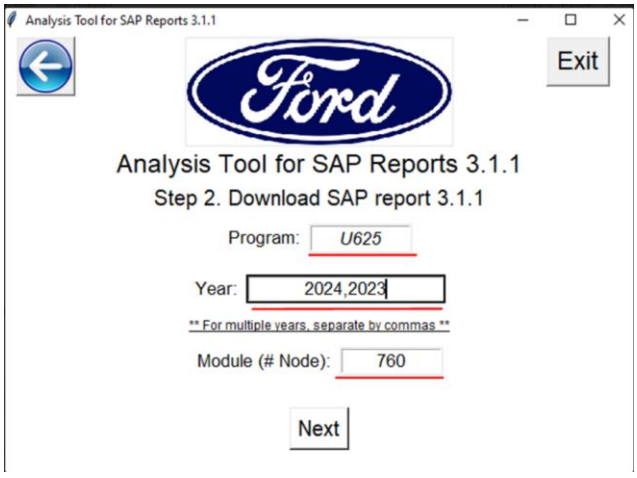

Figura 7. Ventana de la aplicación para ingresar el modelo, año y nodo por analizar

3. Debido a que se requieren los archivos XML para llevar a cabo el análisis, se puede dar el caso en que dichos archivos ya se encuentren descargados, por lo que se le dará la opción al usuario de poder seleccionar la carpeta en donde se ubican dichos archivos. Si se diera el caso en que no se encuentren descargados los archivos, se dirigirá al usuario a una venta en donde podrá seleccionar el rol con el que se desea descargar dichos archivos (aunque la información en general es la misma, existen pequeñas variaciones de información entre cada rol de descarga).
4. Como última entrada del usuario, se le pedirá que indique dónde desea guardar los resultados de la aplicación. La aplicación generará una copia del reporte, ya sea se encuentre descargado o no, en la cual se escribirá la información que se obtuvo a partir del análisis que se realizó para cada vin representativo, y, si se diera el caso de pedir que se descarguen los archivos XML, es la ubicación a donde se moverán dichos archivos.

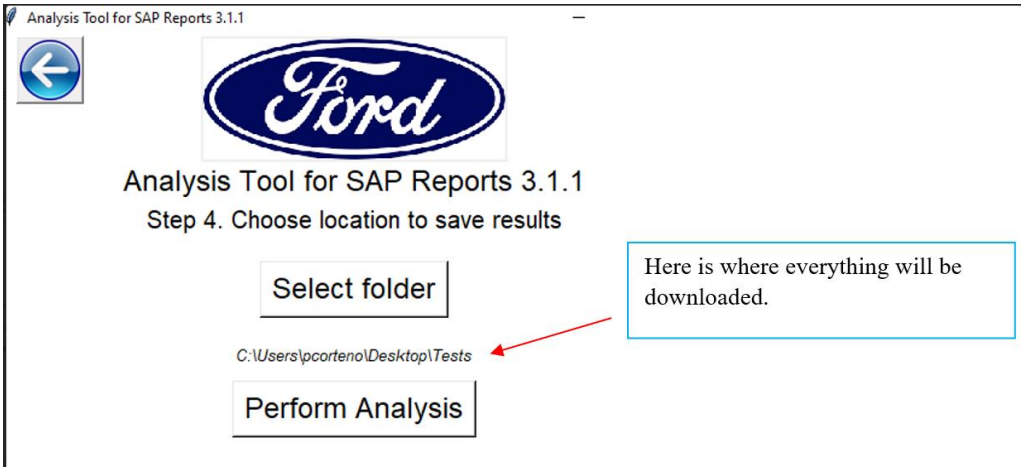

Figura 8. Ventana de la aplicación para seleccionar ubicación de resultados e iniciar análisis

Al terminar de ingresar lo anterior, la aplicación desarrollada tomará la decisión sobre que escenario se deberá ejecutar, tal como se describió anteriormente para después realizar el análisis correspondiente.

Explicación técnica del análisis realizado

En lo explicado anteriormente y en las imágenes mostradas se hace referencia a un análisis que se lleva a cabo, en las siguientes secciones dicho análisis será explicado a continuación, esto consta de diferentes etapas en donde se llevan a cabo descargas de archivos, lectura y extracción de información y escritura de los resultados obtenidos. Al

tener las entradas necesarias y dependiendo del escenario que se tenga, se ejecutarán una o más de las siguientes tareas.

Descarga del reporte a utilizar

Si se entra en alguno de los escenarios en donde no se tiene descargado el reporte, la aplicación abrirá una ventana de un navegador web, para esto se utiliza la librería de Selenium. Esta librería es lo bastante flexible como para darnos la oportunidad de seleccionar el navegador que deseemos utilizar contando como opciones a Google Chrome, Microsoft Edge, etc. Cada uno de ellos cuenta con ciertas ventajas, el punto decisivo fue que la compañía utiliza un tipo de credenciales específico para iniciar sesión con un solo clic en cada una las páginas web de la compañía, así que, para aprovechar este inicio de sesión rápido, se utilizó Microsoft Edge para esta etapa.

La utilización de la librería Selenium nos permite replicar los clics que se realizarían de manera manual, para hacerlo se le debe de indicar al programa cuáles son los elementos a los cuales se les escribirán datos (en el caso de campos de escritura) o a cuáles se les debe dar clic, para esto, a partir del HTML de cada página o ventana web se obtiene el XPATH de cada uno de los elementos con los cuales se interactuará. Es decir, si queremos ingresar una contraseña, se busca el XPATH de dicho elemento y se le indica, mediante código, que dicho elemento recibirá un string. Si bien, cada elemento tiene un XPATH correspondiente, sin embargo, suele suceder que algunos IDs, etiquetas suelan parecerse entre ellas o que estas cambien conforme se ejecuta el programa, cada elemento tiene dos tipos de XPATH, uno corto y otro largo, como los siguientes ejemplos:

- Ejemplo XPATH corto:
    - *“//div/div/button[@title='Refresh menu']”*
    - En este XPATH se está buscando un elemento del tipo botón que tenga por título “Refresh Menu”
- Ejemplo XPATH largo:
    - “/html/body/div[2]/div[2]/div/div[2]/div/section/div/div/div/div[3]/section/div/section/div/div[3]/section/div/div/div/div/div/section/div/section[2]/div/div/div/div/header/div/div/form/input”
    - En este ejemplo, se indica un elemento del tipo input, generalmente utilizado para ingresar datos, como usuario o contraseña

La mayor diferencia que se tiene es la longitud de cada uno, sin embargo, en el segundo caso se tiene una ventaja sobre el otro, como se ha mencionado, a veces, dependiendo de la página en donde se ubique, estos XPATH son dinámicos por lo que suele suceder que los índices tiendan a cambiar por otros números, por ejemplo, si una primer búsqueda realizada se tenía un XPATH de algún elemento, como algún botón, en donde se tuviera "section[2]", suele suceder que para la siguiente iteración este botón podría tener un XPATH diferente con índice diferente, como "section[3]". Por lo que, al percatarse esta diferencia, se optó utilizar XPATH largos para poder realizar iteraciones con ciclos WHILE en donde dicho índice puede ser modificado sabiendo que estos movimientos suelen suceder. Un ejemplo de esto es lo siguiente:

```
i = 13
while True:
    try:
        if driver.find_element(By.XPATH, '//*[@id="__xmlview' + str(i) + '--lovPanelView--search-I"]'):
            driver.find_element(By.XPATH, '//*[@id="__xmlview' + str(i) + '--lovPanelView--search-I"]').send_keys(n)
            break
    except (exceptions.NoSuchElementException, exceptions.StaleElementReferenceException) as e:
        pass
    i = i + 1
sleep(1)
```

Figura 9. Extracto de código mostrando utilización de los XPATH

En la Figura 9, se tiene una muestra de código en donde se inicia un índice "i" con un valor predeterminado debido al estado inicial de la página, después, mediante un ciclo WHILE, se intenta encontrar dicho elemento, en caso de no ser encontrado, al final del código se aumenta en uno el índice descrito, esto sucede hasta que dicho elemento es encontrado.

Este proceso descrito se utilizó dado que las páginas a donde se estaba ingresando no son públicas, sino privadas para la empresa, por lo que no suelen ser estables o tienen tiempos de carga demasiados largos, así que con esto se tiene en consideración los siguientes casos:

- Cuando algún índice del elemento cambie.
- Cuando al primer intento no se encuentre un elemento determinado, utilizando un ciclo WHILE, se hace búsquedas continuas hasta que dicho elemento es encontrado.
- Al ser un proceso que se utilizara frecuentemente, se sabe qué tipo de excepciones son posibles encontrar, así que, mediante estos ciclos infinitos, se ignoran para seguir buscando el elemento

Selección del reporte para analizar

Una de las ventajas que ofrece Tkinter es la interacción con el usuario, para esta situación en específico, la librería de Tkinter tiene una función en donde es posible seleccionar archivos locales, tal como se hace comúnmente en cualquier otra interfaz, para esto, importamos de manera manual un componente de Tkinter llamado “filedialog”, con este es posible realizar la acción descrita, además, de poder delimitar el tipo de archivos que son aceptados a partir de su extensión, como por ejemplo, archivos de Excel (.xlsx), archivos de texto (.txt) o incluso carpetas (*.zip). El resultado que se obtiene no es es el contenido mismo del archivo, sino la ruta de ubicación dentro de la computadora, este facilita el trabajo ya que existen diferentes librerías en Python para procesar los diferentes tipos de archivos.

Extracción de vines representativos del reporte

Al tener el reporte descargado o seleccionado, se debe extraer la información deseada del archivo. Para esto, se utiliza la librería de Python llamada OpenPyXL, con la cual es posible manipular archivos de Excel para escribir, guardar, modificar o borrar el archivo mismo o la información dentro del mismo. Para extraer la información deseada, se planteó la siguiente idea:

Los reportes utilizados en la empresa tienen una estructura estándar, tanto en el formato como en las columnas donde se encuentra la información deseada. En lugar de desarrollar un código dinámico que busque palabras específicas como ‘VIN’ o ‘Total Vehicles’, se seleccionó una muestra de reportes para buscar directamente en las columnas donde se sabe que aparecerá la información.

De esta manera, en lugar de buscar celda por celda el nombre de las columnas del reporte se aprovecha que la información conserva el mismo formato. Esto agiliza la extracción de información, es decir, de los vines representativos.

Se sabe que la información se encuentra en la columna F, de la fila 6 en adelante. Utilizando el componente “load_workbook” de la librería OpenPyXL para leer el archivo, se crean las siguientes líneas.

```
workbook = load_workbook(carpeta_resultados + '\\' + file)
sheet = workbook.active
celda = sheet.cell(2, 3).value
nodo = celda[celda.index('Node') + 6:celda.index('. Module')]

if not sheet['F7'].value == None:
    for row in sheet.iter_rows(min_row=7, max_row=sheet.max_row, min_col=6, max_col=6, values_only=True):
        if row[0] is not None:
            lista_vines.append(row[0])
        else:
            break
else:
    reportes_en_blanco.append(nodo)
workbook.close()
```

Figura 10. Extracto de código para extraer lista de vines representativos

En este pedazo de código se realizan las siguientes tareas:

1. Apertura del archivo.
2. Extracción del nodo descargado (utilizado más adelante).
3. Utilización de una condición IF para el valor de una celda específica para identificar si el reporte está en blanco.
4. Mediante un ciclo FOR utilizando una función de OpenPyXL en donde permite iterar sobre las filas y columnas del archivo, en este caso, estas iteraciones se están delimitando a una columna especifica (F) indicada mediante el puesto correspondiente en el abecedario y un numero de filas que empezara desde la numero 7 hasta donde la función identifique que ya no existe valores. Cada iteración regresará la variable *"row"* y así como el nombre lo describe, esta variable tendrá una lista de valores de las columnas seleccionadas.
5. Dado que se está iterando sobre una lista de un solo elemento, se debe acceder a dicho elemento con el índice 0, se verifica que no sea nulo y se agrega a la lista de vines representativos.
6. El ciclo termina cuando el valor de la lista sea nulo. Se procede a cerrar el archivo para evitar errores en el mismo.

Este proceso se realiza tanto para los casos en donde se descargó el reporte como para el caso donde solamente se seleccionó. Sin embargo, para el primer caso, el usuario ingresa la información de lo que se pretende analizar (modelo, año y nodo). Para el segundo caso, el programa no conoce dicha información, por lo que la obtiene del título del reporte. Como se muestra en la figura 11, el título contiene el modelo (omitido por políticas de

confidencialidad), año y nodo. Mediante procesamiento de strings, se extrae dicha información, con lo que es posible conocer las entradas requeridas sin que el usuario tenga que seleccionar el reporte además de ingresar las entradas.

Model/Year: XXXX/2024. Node: 760. Module: ABS. Total Vehicles: 316,404. Total Collections: 601

Figura 11. Ejemplo del título que contienen los reportes

Descarga de archivos XML con base a los vines representativos

Este paso es probablemente el que más tiempo consume. Para descargar estos archivos, se ingresa a una base de datos privada de la empresa, donde se ingresan 3 datos para obtener el archivo deseado:

1. Vin representativos
2. Rol de descarga (Engineer, Dealer, Consumer, etc.) donde existen diferencias entre el tipo de archivo que se recibe.
3. Nodo deseado. Es posible obtener un archivo con todos los módulos/nodos de un vehículo, sin embargo, dicho archivo es más grande y contiene información innecesaria ya que solamente se desea analizar un solo modulo.

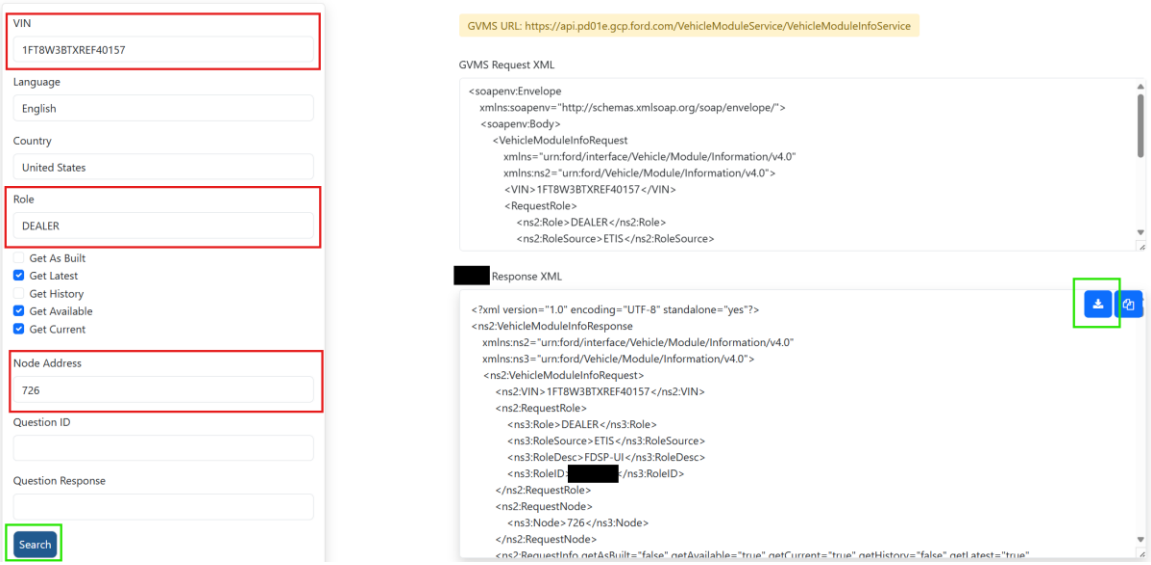

Figura 12. Captura de pantalla de la base de datos utilizada para extraer los archivos XML

Tanto los datos deseados como la forma en la que se ingresó a dicha base de datos se realizan mediante la librería de Selenium, ya que dicha base de datos está alojada en un sitio web (se considera la misma excepción mencionada respecto a la utilización de APIs).

Los pasos que se hacen para la descarga de un solo archivo XML son los siguientes:

1. Escritura del vin representativo.
2. Escritura del rol de descarga.
3. Escritura del nodo por descargar.
4. Realizar búsqueda, indicado en la esquina inferior izquierda de la figura X.
5. Descargar archivo, indicado en la parte media de la página del lado derecho.

La descarga inicial de archivos XML se realizaba de forma secuencial, lo que resultaba ineficiente debido a la variabilidad en la cantidad de VINs y la necesidad de ingresar cada uno individualmente. En los casos con listas de VINs extensas, el proceso podía superar las 2 horas.

Para optimizar este proceso, se implementó una solución basada en multihilos utilizando la librería Multiprocessing de Python. La lista de VINs se dividió en tres partes, y cada parte se asignó a un hilo de ejecución diferente. La elección de tres hilos se basó en un análisis del rendimiento del equipo, buscando un equilibrio entre la velocidad de descarga y la capacidad de realizar otras tareas simultáneamente. Cada hilo ejecutaba la función de descarga de archivos XML, recibiendo como parámetro la sección correspondiente de la lista de VINs. La Figura 13 ilustra la aplicación de multihilos, mostrando la división de la lista de VINs entre los diferentes hilos.

Figura 13. Extracto de código mostrando la utilización de multihilos

Crear carpeta para guardar resultados

Al tener muchos archivos con los cuales se va a trabajar, no es conveniente tener todo en carpetas diferentes, sobre todo si se tiene una gran cantidad de archivos XML. Para tener organizado el resultado obtenido junto con los archivos utilizados para el análisis, se crea una carpeta nueva en una localización que el usuario selecciona.

La creación de dicha carpeta se hace a través de Python, pero no hay una librería con la que se creen carpetas con una sola línea de código. Esto se realiza por la línea de comandos o consola. Sin embargo, Python cuenta con una librería con la cual es posible mandar sentencias a la consola y que se ejecuten sin necesidad de abrir la consola misma.

De esta manera, se solucionaron varias tareas que se llevan a cabo durante el análisis, las cuales son:

- Mover archivos a una localización específica, tanto los reportes como los archivos XML, utilizando el comando move de la consola.
- Renombrar los reportes a un formato específico. Debido a que este tipo de análisis se lleva a cabo de manera constante, es frecuente tener varios reportes en la carpeta de descargas con el mismo nombre. Para distinguirlos, durante el análisis se renombran los reportes con un formato específico: Analysis_"Modelo"_"Año"_"Nodo".

Cuando se ejecuta el proceso, en primera instancia, si se seleccionó el reporte, se extrae la información del modelo, vehículo y nodo. Después, se crea la carpeta de resultados con el formato anterior y ahí se guardan los archivos XML. De la misma forma, si se tiene que descargar el reporte, después de realizar la descarga, se realiza el mismo procedimiento de mover el archivo descargado, así como sus archivos XML correspondientes.

Extracción de información de cada archivo XML

Para realizar este paso, que se realiza con cualquier escenario, se considera la idea mencionada anteriormente: se aprovecha el hecho de que la empresa estandariza lo más posible sus archivos y la información en estos, por lo que se sabe de antemano que la información deseada siempre estará en el mismo lugar.

La estructura que contienen los archivos XML que se analizarán es compleja, por lo que la empresa utiliza namespaces para indicar la estructura. A partir de esto, se buscó y encontró una librería llamada *elementpath*, diseñada para manejar este tipo de archivos. Para utilizarla, se hace uso de una función llamada *"findall",* utilizada para buscar elementos tomando en cuenta el nombre del elemento y el *namespace* en donde se ubica.

```
nivel_1 = xml_data.findall( path: 'ns2:Warning', ns)
nivel_2 = nivel_1[0].findall( path: 'ns3:source', ns)
dicc_aux['Source'] = nivel_2[0].text
```

Figura 14. Extracto de código ejemplificando la navegación en archivos XML

Como muestra la Figura 14, a la variable nivel_1 se le asigna el path del archivo y un diccionario con los namespaces (omitido por confidencialidad). La variable nivel_2 sigue la misma estructura. nivel_1 es una lista porque busca todos los elementos con esa estructura.

Al conocer la ubicación exacta, se evitan ciclos de búsqueda. El archivo tiene una estructura anidada, por lo que se accede a la información por capas (nivel_X), facilitando la navegación.

Los archivos XML contienen atributos y texto. Para algunos criterios, el atributo es importante. Por ejemplo, la presencia del campo "Warning" indica una alerta. La búsqueda de criterios y valores clave optimiza la categorización, deteniéndose si no se encuentra un valor o continuando si se cumplen las condiciones.

Aunque la búsqueda se detenga (en los 3 casos de análisis XML), el diccionario anidado para almacenar la información no queda vacío. Se usan condicionales IF para llenar los campos faltantes con un guion, indicando que el campo no tiene valor. Por ejemplo, si un vehículo no tiene alertas, pero no califica para la Acción de Servicio, los campos "Warning" e "ID SA" tendrán un guion.

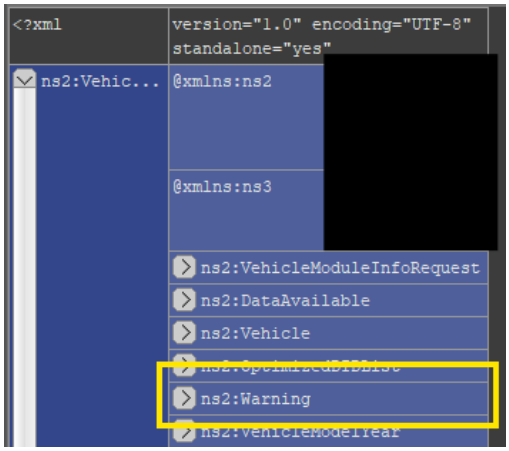

Figura 15. Captura de pantalla ejemplificando la estructura de un archivo XML

De esta forma, es posible obtener la información necesaria. Al final, para obtener algún valor, basta realizar líneas de código semejantes a la última de la figura 15, donde se llama a dicho atributo del elemento que se está llamando.

Escritura de resultados con el formato deseado

El usuario final solo ve el reporte final, aunque la obtención de datos es un proceso complejo que se realiza internamente. El reporte se genera modificando una copia del original con la librería OpenPyXL.

La clave para la organización de la información en el reporte es la estructura de datos utilizada: diccionarios anidados de Python. En lugar de una simple lista, se utiliza una

estructura jerárquica donde cada VIN representativo es una clave, y su valor es otro diccionario que contiene los campos extraídos de los archivos XML. Esta estructura permite una gestión más eficiente y robusta de la información, evitando la ambigüedad y la confusión que podría generar una lista simple.

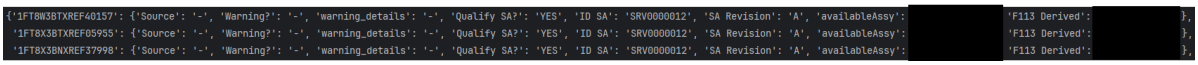

Figura 16. Captura de pantalla ejemplificando la forma en la que se almacena la información extraída de los archivos XML

La existencia de una lista predefinida de VINs representativos agiliza el proceso de análisis. La herramienta recorre esta lista, incorporando la información obtenida de cada VIN directamente en el reporte original. Esto permite tener toda la información relevante en un solo lugar, facilitando su consulta. Los datos se presentan en columnas adyacentes a la información original, respetando el orden preexistente de los VINs para una organización eficiente.

La librería OpenPyXL se utiliza para modificar el reporte, ya que ofrece funcionalidades similares a Microsoft Excel, permitiendo escribir en las celdas, añadir formato y modificar las columnas para una presentación óptima de los resultados. Gracias a la estandarización de los reportes, se conoce de antemano la cantidad de columnas que contiene cada uno. Esto permite escribir la información directamente en las celdas vacías, sin necesidad de recorrer el archivo. Los nombres de las nuevas columnas se escriben inicialmente, y su formato se copia de las columnas originales para mantener la coherencia visual. La escritura de la información se realiza de forma iterativa, tomando como referencia el VIN de cada fila. Finalmente, se guarda el archivo con las modificaciones realizadas. Todo este proceso de escritura se completa en cuestión de segundos.

Resultados obtenidos

Después del proceso mencionado, el resultado entregado al ingeniero a cargo es un documento de Excel, el mismo que se usa como entrada del sistema. Además de las columnas que tiene por default, se agregan columnas extras con la información que cada persona necesita para aplicar a cada uno de los Vin el proceso de prevalidación descrito. Dicha información, obtenida directamente de los archivos XML descargados, cuenta con la información más reciente que se tiene en el sistema, por lo que no existirá un desfase entre la información que analiza el ingeniero y la que se va actualizando en el sistema.

Total Vehicles	Total Assemblies	Vehicle Sample	Strategy	Image	Hardware	Core Assembly	Assembly	Calibration	ECU Configuration

Figura 17. Columnas originales del reporte

Source	Warning?	Qualify SA?	ID SA	SA Revision	F113 Derived	F113 Available	F121 Part Number	Warning Details

Figura 18. Columnas creadas mediante la aplicación para mostrar los resultados

El análisis de cada vehículo resulta en una clasificación en tres categorías: vehículos con alertas, no aptos para la Acción de Servicio, o aptos. Esta clasificación, junto con los detalles y el origen de las alertas, se presenta en la tabla de resultados. La gran ventaja es que, en caso de alertas, la información de los archivos XML permite identificar las causas de manera inmediata.

Antes de la herramienta, la identificación de alertas era un proceso tardío y laborioso, que ocurría durante la validación del siguiente equipo y requería análisis manual. En contraste, ahora, el ingeniero puede realizar una prevalidación completa, identificando y corrigiendo errores antes de enviar el trabajo, lo que mejora la calidad y reduce las alertas.

Si bien el proceso de prevalidación era conocido, su duración lo hacía poco práctico. La implementación de la herramienta superó este obstáculo, ofreciendo una solución automatizada que reduce significativamente el tiempo y el esfuerzo. La ingeniera a cargo de probar la herramienta experimentó una notable disminución en su carga de trabajo, pudiendo realizar el análisis de forma rápida y eficiente.

La utilidad de la aplicación fue evidente para el resto del equipo, quienes también la adoptaron para mejorar sus resultados. Como consecuencia de la notable reducción de alertas, la empresa integró formalmente el proceso de prevalidación en el flujo de trabajo de las Acciones de Servicio. Esto significó que, a partir de ese momento, la prevalidación se convirtió en un paso obligatorio para todos los ingenieros.

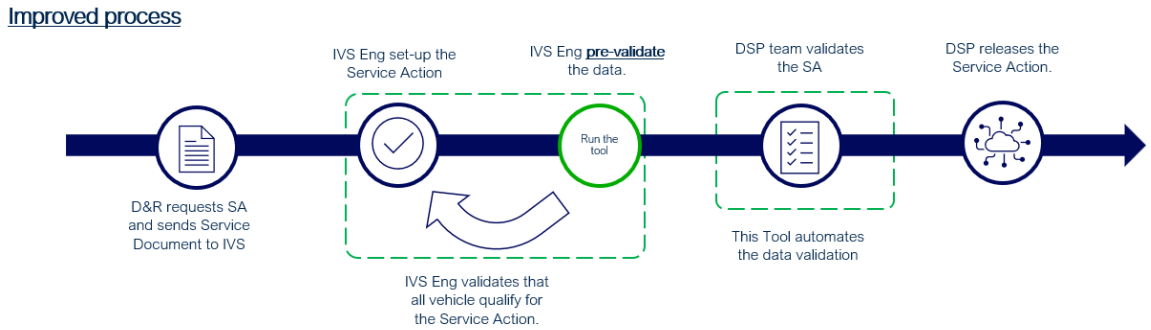

Figura 19. Proceso actualizado con la aplicación incluida

Lo anterior significaba que el ingeniero a cargo entraba en el siguiente ciclo de tareas:

1. Utilización de la herramienta.
2. Encontrar las alertas mediante el reporte generado.
3. Resolver las alertas encontradas.
4. Volver a correr la herramienta con la información actualizada después de arreglar los errores.
5. Buscar en el reporte si persiste alguna alerta.
6. Al no encontrar nuevas alertas, el ingeniero podía pasar su trabajo al siguiente equipo

Dos de los puntos más significativos eran que no solo las personas del equipo que utilizaba la aplicación reducían su carga de trabajo, sino que también aumentaba la calidad de este. Para explicar mejor el segundo, hay que tener en cuenta que los reportes donde se contiene la lista de vines y sus características generales pueden llegar a tener una lista que va desde los 5 vehículos hasta los 400 o, en algunos casos, hasta los 500 vehículos por analizar.

CONCEPTO	Antes	Después
XML análisis (10 VINs)	30 min	6 min
Vehículos que califican	<100%	100%

Tabla 1. Comparación de tiempos antes de la implementación de la aplicación vs después de su implementación

Este número depende de dos variables: el tipo de vehículo (los vehículos más populares o vendidos son de los que habrá más cantidades fabricadas) y el año (los vehículos más viejos, al tener más tiempo en el mercado, es normal que haya más en existencia, siendo lo contrario con modelos no tan populares o que son de años más recientes). Al mencionar que sube la calidad de trabajo entregado, nos referimos específicamente a que, en lugar de solo analizar 10 vehículos en un tiempo considerablemente alto, se tiene la capacidad de analizar toda la lista de vehículos sin importar el tamaño de esta en un menor tiempo.

La empresa realiza un seguimiento de las alertas que surgen durante el proceso de liberación de Acciones de Servicio a través de una plataforma. En esta plataforma, cada alerta se registra como un "Ticket", lo que permite llevar un control preciso del número de alertas activas, su estado actual y el ingeniero asignado para su resolución. La implementación de la herramienta como un paso obligatorio en este proceso tuvo un impacto significativo, no solo en el ahorro de tiempo y esfuerzo para los ingenieros, sino

también en otras áreas. Específicamente, se observó una reducción del 77% en la creación de tickets (de 150,000 a 35,000 por semana) desde que la herramienta se implementó de forma oficial.

Esta reducción se explica porque los ingenieros, al utilizar la herramienta en cada Acción de Servicio, pueden identificar y resolver la mayoría de las alertas en una primera iteración. Esto significa que, cuando el trabajo se envía al siguiente equipo (el de validación), este solo necesita confirmar que los vines están libres de alertas, lo que disminuye considerablemente las devoluciones de Acciones de Servicio y, por consiguiente, la generación de tickets relacionados con estos problemas.

Como se observa en la Figura 20, la cantidad de tickets generados semanalmente ya mostraba una tendencia a la baja antes de que la herramienta se implementara oficialmente en el proceso de validación. Esta tendencia inicial se produjo porque los ingenieros del equipo IVS comenzaron a utilizar la herramienta de forma voluntaria desde el momento en que estuvo disponible.

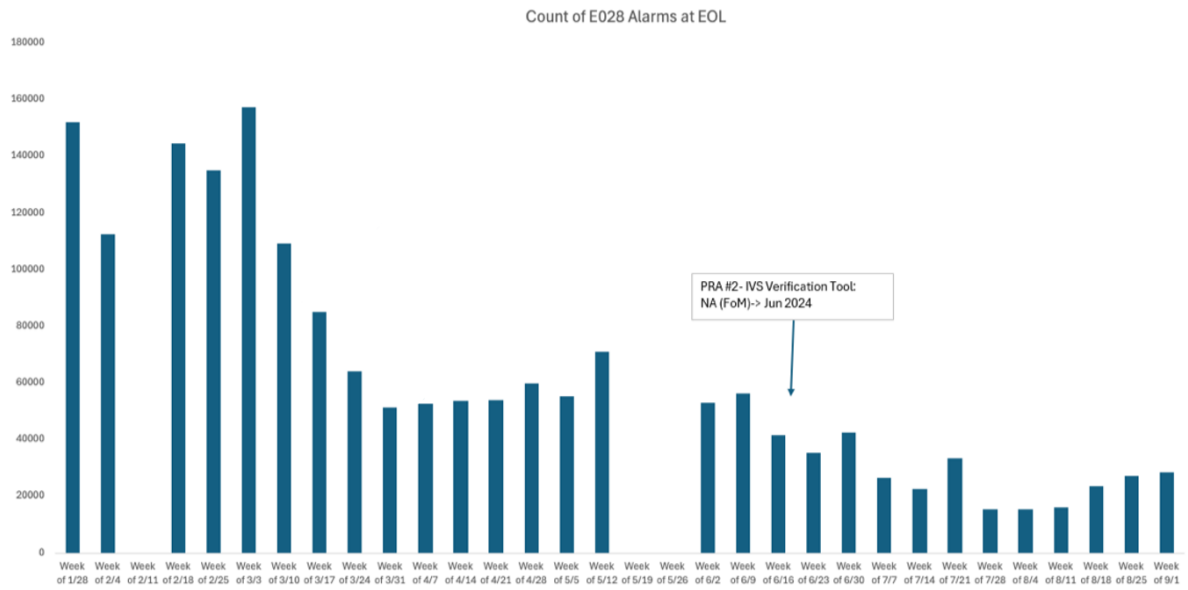

Figura 20. Grafica con la cantidad de tickets creados semanalmente

Los resultados inmediatos y la facilidad de uso de la aplicación impresionaron a los supervisores del área. Esto impulsó la aprobación y el papeleo necesario para que la herramienta se utilizara como parte del proceso de validación en todas las dependencias

de Ford con equipos de IVS a nivel global. Es decir, la herramienta pasó de ser utilizada solo en México a ser implementada en equipos de Australia, China, Europa, India y Turquía.

Este reconocimiento me llevó a realizar videoconferencias con los equipos de estos países para explicar el funcionamiento, las ventajas y el uso de la herramienta. También tuve que crear un manual de usuario que describiera su funcionamiento, los posibles errores y cómo resolverlos. Estas tareas, que inicialmente no formaban parte de mi rol como ingeniero, se volvieron esenciales. Una herramienta, por muy buena que sea, no puede alcanzar su máximo impacto si los usuarios no la conocen y no saben cómo utilizarla correctamente.

Actualmente, las variaciones en cuanto a la cantidad de tickets creados derivados de este tipo de alertas no han aumentado ni disminuido. Esto se debe a que llegó a un estado de estabilidad en cuanto a la generación de las alertas, originando que la reducción de tickets que podía darse del lado del equipo de IVS haya alcanzado un punto en donde el equipo ya no puede influir de manera significativa en este número de tickets dado que se estaría invadiendo el trabajo de otros equipos. Gracias a la reducción de carga de trabajo de los ingenieros de IVS, se pueden enfocar en solucionar los problemas los cuales podrían impactar al usuario final.

Conclusiones

En mi tiempo en Ford Motor Company, he participado en diversos proyectos. Estos me han demostrado que no todos los conocimientos específicos de mi carrera se aplican directamente en mi día a día laboral. Principalmente, esto se debe a que el área en la que trabajo difiere en cierta medida de lo que estudié.

A pesar de esto, la formación que recibí me proporcionó una valiosa habilidad: la capacidad de analizar y resolver problemas. Esta habilidad me permite adaptarme a las diferentes situaciones que se presentan y ofrecer soluciones ante los desafíos. En consecuencia, me considero un profesional flexible, capaz de involucrarme en diversos ámbitos dentro de la empresa.

Mi carrera, Ingeniería Mecatrónica, se caracteriza por la sinergia de cuatro disciplinas: mecánica, electrónica, industrial y computación. Esta combinación me ha convertido en una persona capaz de analizar un mismo problema desde diferentes perspectivas, considerando cada una de las ingenierías que conforman la mecatrónica. Aplicando esto a mi trabajo, puedo encontrar soluciones diversas, ya que mi forma de pensar está entrenada para abordar las problemáticas desde diferentes ángulos.

Por otro lado, al trabajar en una empresa que se mantiene a la vanguardia de los desarrollos tecnológicos e industriales, he notado que los conocimientos impartidos en mi carrera, si bien son una base sólida, requieren una actualización constante para mantenerse a la par de las nuevas tecnologías. Esto significa que, además de aprender las normas y procesos internos de la empresa, he tenido que estudiar nuevas tecnologías que, aunque conocía teóricamente, no había utilizado en la práctica. Esta necesidad de actualización implicó una curva de aprendizaje acelerada para poder cumplir con los objetivos establecidos.

Además, existe la idea de que los ingenieros, especialmente aquellos de carreras más orientadas a la industria, tienen poco contacto con clientes o con personas del área administrativa. Sin embargo, mi experiencia en Ford ha demostrado que la realidad es diferente. Desde mi llegada a la empresa, he estado involucrado con personas a las cuales se les presentan avances, progresos, soluciones y obstáculos que surgen en los proyectos.

En muchas ocasiones, me he visto en la necesidad de explicar conceptos técnicos complejos de manera sencilla y directa, con el fin de comunicar el estado de un proyecto de forma concisa. Al principio, esto representó un desafío, ya que durante la carrera no tuve la necesidad de realizar presentaciones de resultados como se hacen en el ámbito laboral.

Esto me llevó a desarrollar nuevas habilidades, aprendiendo a exponer la información de forma profesional a personas con un alto nivel corporativo, mostrando tanto los aspectos positivos como los desafíos que se presentan en el desarrollo de un proyecto.

En resumen, gracias a mi carrera, he desarrollado la capacidad de adaptarme a diferentes situaciones, aprender de forma continua y superar las adversidades. Si bien la institución educativa debe seguir actualizando sus planes de estudio, la formación que recibí me brindó habilidades fundamentales que son útiles en el campo laboral. Estas habilidades, como la flexibilidad y la capacidad de análisis, no se adquieren fácilmente, sino que requieren años de práctica, donde la universidad juega un papel esencial.

Referencias

[1] Ford Motor Company, "Acerca de Ford," [Online]. Available: https://www.ford.com.co/about-ford/ford-motor-company/.

[2] Grupo Flores, "La historia Ford," [Online]. Available: https://www.dimasaford.com/historia-ford/.

[3] M. Vázquez, "FORD MEDIA CENTER," 26 8 2020. [Online]. Available: https://media.ford.com/content/fordmedia/fna/mx/es/news/2020/08/26/hace-95-anos-se-inauguro-la-primera-planta-automotriz-en-mexico.html.

[4] Ford Motor Company, "Así funciona el GTBC, el Centro de Ingeniería Ford más Grande de México," [Online]. Available: https://www.ford.mx/blog/innovacion/gtbc-centro-ingenieria-mexico-sep2023/.

[5] FORD MEDIA CENTER, "Ford de México traslada sus operaciones al Centro Global de Tecnología y Negocios México," 12 09 2022. [Online]. Available: https://media.ford.com/content/fordmedia/fna/mx/es/news/2022/09/12/ford-de-mexico-traslada-sus-operaciones-al-centro-global-de-tecn.html.

[6] I. Juárez, "Ford da la bienvenida a la 4ta generación de PD Academy," 6 8 2024. [Online]. Available: https://media.ford.com/content/fordmedia/fna/mx/es/news/2024/08/06/ford-da-la-bienvenida-a-la-4ta-generacion-de-pd-academy.html.

[7] Amazon Web Services, "¿Qué es una interfaz de programación de aplicaciones (API)?," [Online]. Available: https://aws.amazon.com/es/what-is/api/.