Trabajo Fin de Grado
Ingeniería Aeroespacial

Automatización en la Detección y Prevención de errores en Publicaciones Técnicas aeronáuticas.

Autor: José Manuel Trujillo Rodríguez
Tutor: Laura García Ruesgas

Dpto. Ingeniería Gráfica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Proyecto Fin de Grado
Ingeniería de Aeroespacial

Automatización en la Detección y Prevención de errores en Publicaciones Técnicas aeronáuticas.

Autor:
José Manuel Trujillo Rodríguez

Tutor:
Laura García Ruesgas
Profesora colaboradora

Dpto. Ingeniería Gráfica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Grado: Automatización en la Detección y Prevención de errores en Publicaciones Técnicas aeronáuticas.

Autor: José Manuel Trujillo Rodríguez

Tutor: Laura García Ruesgas

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mis padres

Agradecimientos

Agradecer en primer lugar a mi tutora Laura García por su flexibilidad y capacidad de adaptar este trabajo a mis necesidades académicas y profesionales. Por su guía, su ayuda y sus propuestas.

A mis padres por sacar de mí más de lo que se podía esperar. A mis hermanos por ser mi apoyo durante estos duros años de grado. A mis abuelos por el cariño y cuidado que me han dado.

Quiero agradecer también a quienes se cruzaron en mi vida académica para aportarme personalmente, mis amigos Don Daniel, Isabel y Salvi.

Por último, a mis compañeros de grado que se convirtieron en amigos.

José Manuel Trujillo Rodríguez
Estudiante Grado Ingeniería Aeroespacial
Sevilla, 2019

Resumen

El objetivo de este Trabajo de Fin de Grado es la automatización de la detección y prevención de determinados errores en las Publicaciones Técnicas (PPTT) aeronáuticas de la empresa Airbus Defence and Space y concretamente para el programa militar A330-MRTT.

Las PPTT contienen algunos errores debido a que sobre ellas trabajan una gran cantidad de personas con grados de experiencia y conocimientos muy diversos. En ocasiones no se logra validar de manera exhaustiva todo el trabajo y esto puede conllevar a un error en la publicación de un manual. Si tenemos en cuenta la enorme cantidad de datos que se editan/incluyen en cada revisión (cada 3 meses) de cada manual y el tiempo que lleva desarrollándose este programa (más de una década) podemos prever que el número de errores no será despreciable: aunque no lo sea en valor relativo a la información total de los manuales sí que lo es en valor absoluto.

Estos errores pueden traer consecuencias negativas al departamento implicado en su producción. Por ello, debido al actual estado de madurez del proyecto, se está comenzando a focalizar esfuerzos por detectar y resolver estos problemas. El gran obstáculo para lograr este nuevo objetivo es la falta de perfiles en la empresa con conocimientos de programación capaces de desarrollar códigos que puedan detectar dichos errores. Debido a la gran cantidad de información que se maneja no es asumible la detección de errores de manera manual con el tiempo y el número de personas que trabajan en el proyecto.

El objetivo de este trabajo es trazar de manera accesible, rápida y automática estos errores sin que requiera el trabajo de uno o varios trabajadores durante varios días o meses, solventando así el problema descrito. Para ello, se automatizarán algunos casos de modo que sirva para sentar las bases de cómo debería hacerse para el resto de casos.

Adicionalmente, se propone un modelo de generación automática de un manual para así reducir la posibilidad de cometer un error humano en las publicaciones.

Abstract

The aim of this End of Degree Project is the automation of the detection and prevention of certain errors in the aeronautical Technical Publications (PPTT) of the company Airbus Defence and Space and specifically for the military programme A330- MRTT.

The PPTTs contain some errors because a large number of people with very different degrees of experience and knowledge work on them. Sometimes it is not possible to validate all the work exhaustively and this can lead to an error in the publication of a manual. If we consider the enormous amount of data that is edited/included in each revision (every 3 months) of each manual and the time that this program has been developing (more than a decade) we can foresee that the number of errors will not be negligible: although it is not in relative value to the total information of the manuals it is in absolute value.

These errors can have negative consequences for the department involved in their production. Therefore, due to the current state of maturity of the project, efforts are beginning to focus on detecting and solving these problems. The big obstacle to achieve this new goal is the lack of profiles in the company with programming knowledge capable of developing codes that can detect such errors.

Due to the large amount of information handled it is not possible to detect manually errors with the time and the number of people working on the project.

The aim of this project is to outline these errors in an accessible, fast and automatic way without requiring the work of one or some workers for several days or months, thus solving the problem described. To this end, some cases will be automated so that it serves to lay the foundations of how it should be done for the rest of cases.

In addition, an automatic generation model of a manual is proposed to reduce the possibility of human errors in publications.

Índice

Notación

ADL	Allowable Damage Limit
ADS	Airbus Defence And Space
AMM	Aircraft Maintenance Manual
AOG	Aircraft On Ground
ASM	Aircraft Schematics Manual
AWL	Aircraft Wiring List
AWM	Aircraft Wiring Manual
BITE	Built-In Test Equipment
CM	Conversion Manual
CMM	Customer Maintenance Manual
DLL	Dynamic Link Library
DTD	Document Type Definition
ELA	Electrical Load Analysis
ERC	Engineering Request for Change
FOXE	First Object XML Editor
IPC	Illustrated Parts Catalogue
MMEL	Master Minimum Equipment List
MPD	Maintenance Planning Document
MRCM	Mission Role Change Manual
MRTT	Multi Role Tanker Transport
PPTT	Publicaciones Técnicas
RDL	Repair Damage Limit
RFC	Request For Customer
SGML	Standard Generalized Markup Language
SRM	Structural Repair Manual
TR	Technical Request
TSM	Trouble Shooting Manual
UARRSI	Universal Aerial Refueling Receptacle System Installation
WO	Work Order

1. Introducción

El sector de las Publicaciones Técnicas aeronáuticas es uno de los campos en los que menos se ha invertido tradicionalmente, lo que provoca que actualmente la producción de los mismos se realice en ocasiones mediante herramientas desfasadas y de complejo manejo. Generalmente, los cambios o revoluciones se realizan por petición de un cliente que está dispuesto a pagar una cantidad elevada de dinero con tal de que la empresa desarrolle nuevas herramientas. Gracias a estos proyectos, se invierte y los nuevos resultados se extrapolan posteriormente al resto de programas.

Dentro de las Publicaciones Técnicas, distinguimos diversos niveles en cuanto a la generación del trabajo. Un primer nivel sería la autoría técnica, que se corresponde con las labores encargadas de interpretar la documentación de Diseño y la generación de las tareas/figuras o actualización de las mismas de acuerdo a dicha documentación.

Figura 1.1: Niveles en las PPTT

Como segundo nivel, encontraríamos la validación por parte de expertos de la correcta realización del trabajo de los autores técnicos.

El nivel de Clean-up tiene como objetivo tomar toda la información de uno o varios manuales y detectar y resolver todos los posibles errores que violen el correcto funcionamiento de la Publicación Técnica o manual, es decir, errores que no sólo pueden darse por una mala interpretación de la documentación por parte de un autor técnico, sino también aquellos errores derivados de una incorrecta implementación de una información bien interpretada.

Por último, la gestión de la producción debe encargarse de la coordinación de todos los niveles anteriores.

Este TFG queda abarcado dentro del nivel Clean-up. Este nivel ha surgido recientemente como una necesidad para alcanzar los valores deseados de calidad en los manuales, con clientes cada vez más exigentes y con una cantidad de información a tratar cada vez más elevada.

En los recientes inicios del Clean-up se ha llevado a cabo la trazabilidad de los errores de manera manual. Esta ejecución ha sido muy susceptible de error humano y al mismo

tiempo inabarcable con la mano de obra que se dispone. Este TFG trata de automatizar estas labores, de manera que el error humano quede relegado al programador en el desarrollo de su código y de modo que puedan trazarse en cuestión de minutos todos los errores en los que se esté interesado.

1.1. La empresa y el avión

Los conceptos aplicados a este TFG son fácilmente extrapolables a otras empresas y dentro de la misma, a otros proyectos. No obstante, vamos a particularizar el proyecto desarrollado para la empresa *Airbus Defence and Space* y el programa *A330-MRTT*.

Airbus es la corporación industrial más importante de la Unión Europea dentro del segmento de la aviación y espacio y el mayor grupo aeroespacial del mundo, disputándose el primer puesto con la empresa estadounidense Boeing. Segúnse define la propia empresa, como líder del sector aeroespacial mundial, Airbus diseña, produce y entrega soluciones innovadoras con el objetivo de crear un mundo mejor conectado, más seguro y más próspero [1].

Esta corporación se divide a su vez en 3 divisiones:

- **Airbus Civil**. Es la división que más ingresos reporta a la empresa y es la encargada del desarrollo y fabricación de los aviones civiles de la marca.
- **Airbus Defence and Space (ADS)**. Es la encargada del desarrollo y fabricación de sistemas tácticos, aeronaves tanqueras y aviones de combate avanzados. Actualmente, ADS desarrolla las aeronaves A400M, C-295, C-235, C-212, Eurofighter Typhoon y A330-MRTT.
- **Airbus Helicopters**. División que desarrolla y fabrica helicópteros tanto civiles como militares.

El Airbus A330-MRTT es un avión basado en el Airbus A330-200 de transporte de pasajeros. Airbus Defence and Space compra a Airbus Civil el avión de pasajeros mencionado (que recibe la designación de *avión verde* dentro de ADS, por ser el color que tiene la aeronave cuando la reciben) y le realiza una serie de modificaciones para que la aeronave pueda garantizar las actuaciones y características militares deseadas.

La designación MRTT (Multi Role Tanquer Transport) es explicativa del sentido de la concepción de esta aeronave. Puede servir como avión cisterna para otros aviones, tiene la capacidad de recibir fuel en algunas de sus versiones (mediante el UARRSI - Universal Aereal Refueling Receptacle System Installation), puede tener configuración médica, de transporte de pasajeros y de transporte de vehículos pesados.

La empresa define la aeronave como el tanquero de nueva generación con mayor capacidad, probado en combate y con capacidades de multi-rol únicas, convirtiéndose el A330-MRTT (basado en el avión civil A330-200) en el avión de referencia para los tanqueros de multi-rol [2].

1.2. El sector de las Publicaciones Técnicas (PPTT)

Las publicaciones técnicas de una aeronave son el conjunto de documentos y servicios derivados de los mismos que se entregan junto con la aeronave implicada y son necesarios para el mantenimiento, correcto uso e identificación de los diferentes equipos, sistemas y

componentes estructurales y estéticos que componen la aeronave.

El Airbus A330-MRTT se diseña y transforma en la planta de Airbus de Getafe, pero las publicaciones técnicas de mantenimiento del mismo, así como las relativas al training de los pilotos, se elaboran en la planta de San Pablo, en Sevilla.

Desde Getafe envían los planos y otros documentos con los que se desarrolla la aeronave al departamento de Publicacion Técnicas del A330-MRTT (TASSD2) y se utilizan para elaborar los manuales de mantenimiento de la aeronave.

La documentación enviada queda englobada mediante diversos criterios, según sea la naturaleza de la modificación que hay que implementar en la aeronave. Distinguiremos los siguientes criterios:

- **Modificación**. Un conjunto de planos cuya implementación tiene un objetivo común. Es frecuente encontrarse modificaciones de preconfiguración de la aeronave, que generalmente consta de muchos planos derivados de los empleados en otros aviones (en ocasiones incluso reutilizados) que se utilizan como base de partida o conjunto de modificaciones básicas a partir de las cuales se va modificando el avión para que tenga las características específicas que lo distinguen del resto de A330-MRTT. No obstante, podemos encontrarnos modificaciones con objetivos menores, como por ejemplo una modificación que pudiera estar formada por 12 planos que tienen como objetivo redireccionar unas tuberías de líquido refrigerante a lo largo de todo el avión porque el cliente ha querido añadir a su aeronave un sistema que impide, por su localización en la aeronave, que se utilice el sistema tradicional con el que se había estado trabajando.

- **ERC (Engineering Request for Change)**. Se trata de uno o varios planos que reflejan un cambio propuesto generalmente por el operario que estaba montando una parte determinada del avión y ve inviable poder ejecutar su labor siguiendo las directrices de los planos de ingeniería. Por ejemplo, el operario trata de instalar un pequeño bracket en una zona determinada pero choca con otro elemento. Entonces, éste decide realizar un pequeño corte al bracket que no influya en su función. El operario elabora un documento que los ingenieros de diseño revisan y, en caso de que den el visto bueno, se envía una ERC que recoge los cambios efectuados. En caso de que el cambio propuesto por el operario no se acepte por parte de ingeniería, éstos elaboran una nueva solución y se expide igualmente una ERC pero con los cambios desarrollados por ingeniería.

- **Concesión**. Se trata de una leve modificación en la aeronave por necesidad en el proceso de montaje, al igual que sucedía con la ERC, con la salvedad de que no necesita aprobación por parte del equipo de ingenieros.

- **Boletín de servicio**. Los boletines de servicio recogen cambios que deben efectuarse sobre la aeronave pero una vez que ésta ya ha sido entregada al cliente. Un ejemplo son los boletines de servicio para la implementación de equipos militares con los que no se permiten a las aeronaves sobrevolar el espacio aéreo español. Si a un cliente se le entrega el avión en Getafe y tiene que ir volando hasta su país, no puede llevar el equipamiento montado en la aeronave, por lo que debe esperar a llegar a su país y, utilizando el boletín de servicio, podrá montar el equipo.

1.3. Los manuales

Las publicaciones técnicas de una aeronave se explican a continuación. Destacar previamente que la nomenclatura que se empleará en este trabajo sigue las directrices del A330-MRTT. A modo de ejemplo, el manual de mantenimiento de estructuras de la aeronave recibe el nombre de SRM (Structural Repair Manual) para el A330-MRTT, mientras que recibe el nombre de ASR (Aircraft Structural Repair) para el caso del A400M.

- **AMM (Aicraft Maintenance Manual)**. Se trata del manual (o publicación técnica) de mayor complejidad y carga de trabajo por lo que implica a un mayor número de trabajadores en su desarrollo. Este manual muestra las tareas necesarias para el mantenimiento de un equipo o sistema del avión que debe realizarse sobre el mismo avión y no en el taller. Incluye la explicación de cómo ejecutar pruebas BITE (Built-In Test Equipment), medida de magnitudes eléctricas de equipos, instalación y desinstalación de equipos, comprobación de la interconexión entre sistemas...

- **TSM (Trouble Shooting Manual)** El TSM recoge los procedimientos a seguir cuando sucede un fallo en la aeronave. De manera algorítmica va guiando al operario, indicándole pruebas a realizar para detectar fallos y propone nuevas pruebas en función de los resultados de la anterior.

- **IPC (Illustrated Parts Catalogue)** Este manual se emplea para la identificación de los equipos instalados en la aeronave y cuyo mantenimiento se realiza sobre la misma. Trabaja de manera conjunta con el AMM: el AMM explica las tareas a realizar sobre un equipo, mientras que el IPC identifica dónde están estos equipos en la aeronave y da información sobre los mismos, como por ejemplo sus Part Number.

- **SRM (Structural Repair Manual)** El SRM está dividido a su vez en dos partes. *Identificación*: Tiene como objetivo la identificación de los componentes estructurales de la aeronave susceptibles de sufrir daños que afecten a la seguridad en la navegabilidad aérea de la aeronave. Incluye desde los grandes paneles que recubren el fuselaje de la aeronave, pasando por las vigas, largueros y larguerillos hasta pequeños brackets situados en zonas críticas y desempeñando una función importante en la estructura de la aeronave.
  ADL/RDL o *Allowable Damage Limit / Repair Damage Limit*: Recogen, para un elemento estructural determinado, el máximo de daños admisibles por el mismo (profundidad de una grieta o largo, grado de corrosión, plastificación,...), los procedimientos a emplear para poder evaluar dichos daños y cómo llevar a cabo las reparaciones en caso de que sea posible.

- **ASM (Aircraft Schematic Manual), AWM (Aircraft Wiring Manual), AWL (Aircraft Wiring List)**: El propósito de estos manuales es permitir un conocimiento completo de los sistemas eléctricos, electrónicos y electromecánicos, comprendiendo la instalación de los mismos y sus conexiones físicas. Por un lado, el ASM provee una visión funcional de sistemas y subsistemas que estén activados en la aeronave. Por otro lado, el AWM proporciona una visión física de estos sistemas y subsistemas, de modo que se defina la instalación de los mismos. Por último, el AWL refleja la lista de cables presentes en cada aeronave.

- **CMM (Customer Maintenance Manual)**: Del mismo modo que el AMM se emplea para el mantenimiento de equipos montados sobre la aeronave y cómo ejecutar los test de comprobación de estado de los mismos y se usa en conjunción con el IPC para identificar dichos equipos y piezas que los componen, el CMM es un

manual que contiene la información de ambos pero exclusivamente para equipos que no han sido desarrollados por Airbus, sino que han sido comprados a un *vendor*, que es como se les conoce habitualmente a los proveedores de equipos externos a la empresa.

- **Manuales pequeños**: Existen numerosos manuales además de los ya descritos. No obstante, estos manuales carecen del nivel de importancia que tienen los otros y por ello, generalmente no se desarrollan nuevas versiones en cada revisión para cada cliente. Uno de ellos es el ELA (Electrical Load Analysis), que contiene información sobre qué equipos pueden conectarse a qué buses del avión para poder obtener la potencia deseada en dicho equipo y no interferir en el correcto funcionamiento de otros.

  Concretamente el ELA es un manual que no se actualiza si llegan nuevos criterios, sino que contiene una descripción en la que se explica cómo debe actualizar cada cliente su manual en caso de que haya cambios en su avión. La información que contiene el manual debe estar de acorde al estado del avión en el momento de su entrega al cliente. No obstante, que no se actualicen los manuales de acuerdo a los nuevos criterios que le afecten no significa que no sea necesario editarlos por cambios generales en las PPTT.

  El ELA es un manual desarrollado usando el conocido software de Microsoft: Word. Existe un manual para cada avión de cada cliente y cada vez que es necesario llevar a cabo un cambio en el manual, hay que hacerlo generalmente para cada avión y hay que actualizar absolutamente todas las hojas de dicho manual. A veces hay que actualizar el logo, cambiar la introducción, incluir información que nunca se incluyó pero se empieza a estimar que debería aparecer...

1.4. Particularidades del programa A330 MRTT (programa híbrido)

Trabajar en la PPTT de un programa híbrido tiene una dificultad añadida sobre programas puramente militares. En los programas militares, los trabajadores de las PPTT tienen acceso a toda la información importante para desempeñar su trabajo de la aeronave sin restriccciones. Además, toda esta información se maneja utilizando un software específico y toda la información de sus manuales ha sido incluida por ellos mismos.

Por el contrario, el Airbus A330-MRTT compone sus publicaciones técnicas de información proveniente de las modificaciones realizadas sobre la aeronave para militarizarla y, por otro lado, hay bastante información sobre el avión civil del que parte que debe mantenerse. Además, para manejar la información del avión verde se utilizan herramientas específicas de Airbus Civil: del mismo modo que un cliente no tiene el mismo acceso a la información de los manuales del MRTT que los trabajadores, los trabajadores del MRTT (que son clientes para Airbus Civil) no pueden acceder a toda la información del avión verde. Asimismo, hay que tener una compleja trazabilidad entre las tareas que conviven en el manual siendo completamente idénticas a las del avión civil, las que son híbridas (tareas civiles editadas) y las que son exclusivamente del programa militar.

Por esto, según el manual del que hablemos, tendremos figuras/tareas pertenecientes 100 % a avión verde (tendrán atributo $AIL\text{-}REV = \text{UC}$); integradas, que son las editadas

pero manteniendo alguna información de avión verde (*AIL-REV* = RC); y las completamente militares (*AIL-REV* = NC).

Ejemplo de tareas:

- **UC.** Tarea de mantenimiento sobre las luces del tren de aterrizaje: el tren de aterrizaje es una de las partes del A330 MRTT que más se ha podido aprovechar de la versión civil de la que procede la aeronave.
- **RC.** Tarea de inspección sobre el ala: una de las modificaciones que sufre el avión militar es el montaje de winglets: extensiones verticales de las puntas del ala que mejoran la eficiencia y alcance de la aeronave [3]. Por tanto, tareas como la de inspección del ala pueden aprovecharse en su mayoría añadiéndole subtareas adicionales que comprendan las inspección de los winglets.
- **NC.** Cualquier tarea asociada a la inspección, montaje/desmontaje o test de un equipo puramente militar, como por ejemplo el BOOM.

1.5. Uso de las Publicaciones Técnicas en una aeronave

Los manuales no se utilizan porque sí, sino que siempre hay un motivo para utilizarlos. Estos motivos o entradas son diversos. Un vez que se ha acudido a un manual por una entrada determinada, ya pueden entrelazarse el resto de manuales mediante llamadas los unos a los otros. Las diversas entradas o motivos por los que se decide acudir a las PPTT de una aeronave son:

- **MPD (Maintenance Planning Document).** Este documento detalla la frecuencia con la que deben realizarse tareas de diversas índoles, como inspección o tests, a un equipo, sistema o estructura determinados. Esta frecuencia puede darse por horas reales, horas de vuelo o ciclos de vuelo.
- **TSM (Trouble Shooting Manual).** Cuando vemos que algo no funciona bien en la aeronave o nos salta un aviso de error, acudimos a este manual para tratar de identificar el fallo y que nos redirija a otro manual para tratar de resolverlo.
- **MRCM (Mission Role Change Manual)**. Cuando queremos cambiar de rol a la aeronave. Por ejemplo, queremos quitarle los PODS que actualmente tiene equipados. Pese a tener sus propias tareas el MRCM, también redirige en otras ocasiones a otros manuales.
- **MMEL (Master Minimum Equipment List).** Se acude a este manual generalmente cuando tenemos un equipo averiado y queremos saber si podemos volar sin dicho equipo.
- **CM (Conversion Manual).** En el caso del MRTT, aplica únicamente a la flota FSTA, puesto que son las únicas aeronaves capaces de volar con rol civil o militar, a elección. Este manual recoge las tareas necesarias para transformar la aeronave de un rol a otro.
- **Tareas no-programadas.** Se encuentran en el ATA 05 de AMM. Acudes a este tipo de tareas cuando has tenido un altercado con la aeronave y quieres comprobar si ha habido daños en la misma. Por ejemplo, el impacto en el aterrizaje ha sido muy fuerte o ha caído un rayo en el avión y quieres comprobar si alguna estructura ha sido dañada.

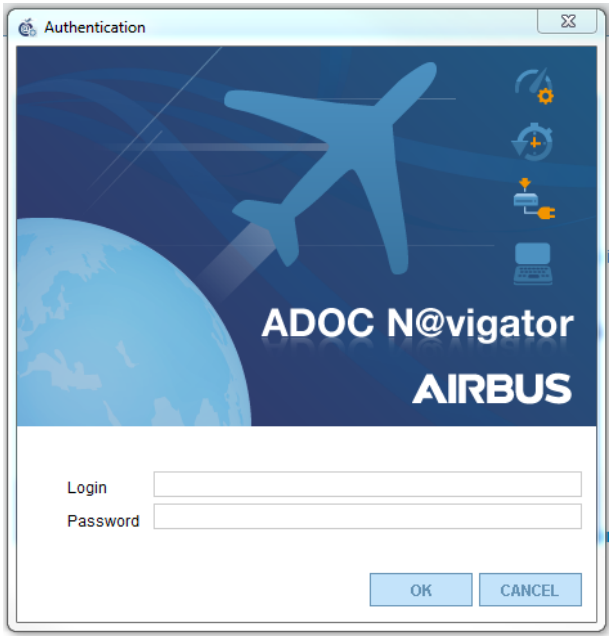


Figura 1.2: Ventana de acceso a Airn@v

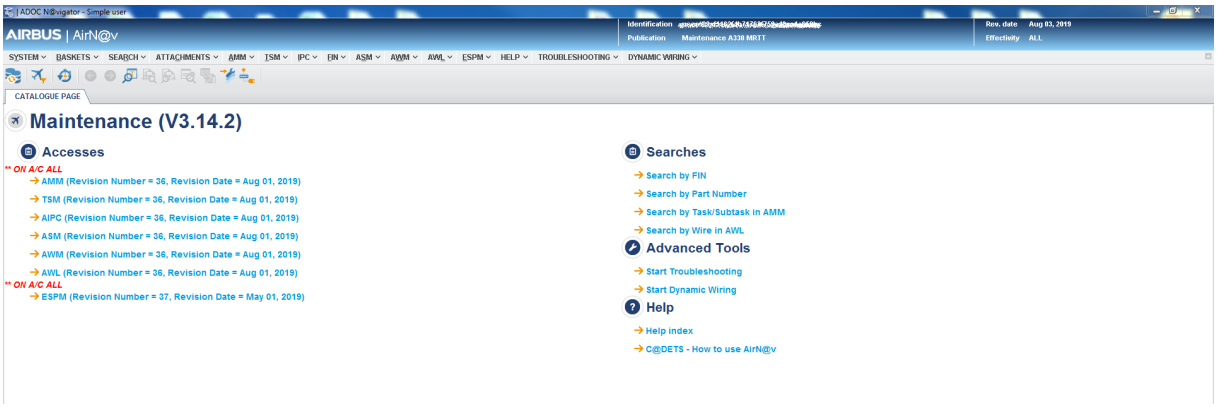


Figura 1.3: Ventana principal de acceso a Maintenance en Airn@v

Se podrá acceder a los principales manuales mediante la herramienta *Airn@v* que se le entrega al cliente.

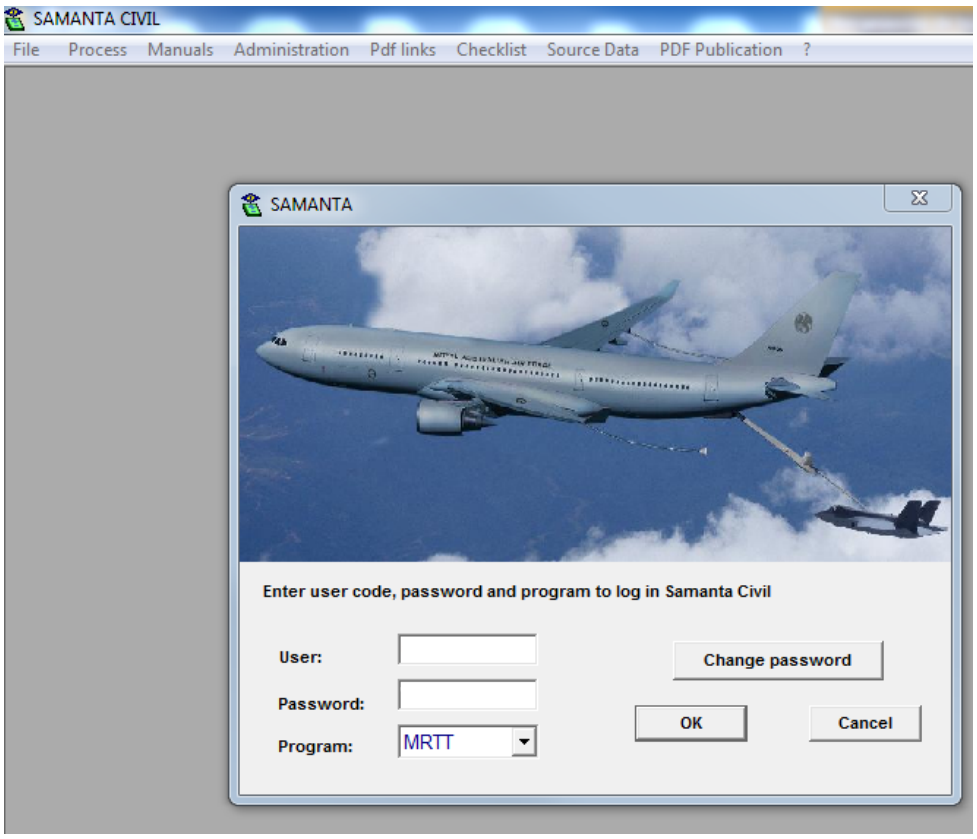


Figura 2.1: Ventana de acceso a Samanta Civil

2. Softwares, lenguajes y normas empleadas en las PPTT

El mundo de las Publicaciones Técnicas está muy poco automatizado y se emplean numerosas herramientas que no necesariamente guardan consistencia entre sí. Si bien para los manuales más importantes sí que se han desarrollado plataformas que faciliten el trabajo de los autores técnicos, para el resto de publicaciones hay que hacer labores muy manuales para poder entregarlos.

2.1. Softwares

Actualmente, para el programa MRTT, se emplea la herramienta de *Samanta Civil* para trabajar en los manuales AMM, SRM, AWM, ASM, AWL y TSM de modo que los autores técnicos puedan trabajar con las unidades mínimas de producción de cada manual (los PageBlocks o tópicos). De este modo se puede trabajar de manera controlada sobre los manuales y se tiene trazado quién entrega qué documento en qué momento y quién lo valida. A través de este software se puede acceder a las unidades mínimas de producción abriéndolas desde *Arbortext Editor*, el cual da acceso a la edición por parte de los equipos de autoría permitiendo la colaboración en proyectos de documentación grandes y complejos que necesitan ser difundidos de diversas formas [4].

Este software se utiliza también para la asociación de tópicos a las Work Orders (WOs), que son las unidades de trabajo empleadas para distribuir el trabajo entre las empresas subcontratadas y evaluar la calidad de sus trabajos. Las WOs tienen una descripción donde debe explicarse el trabajo que va a desarrollarse bajo la misma.

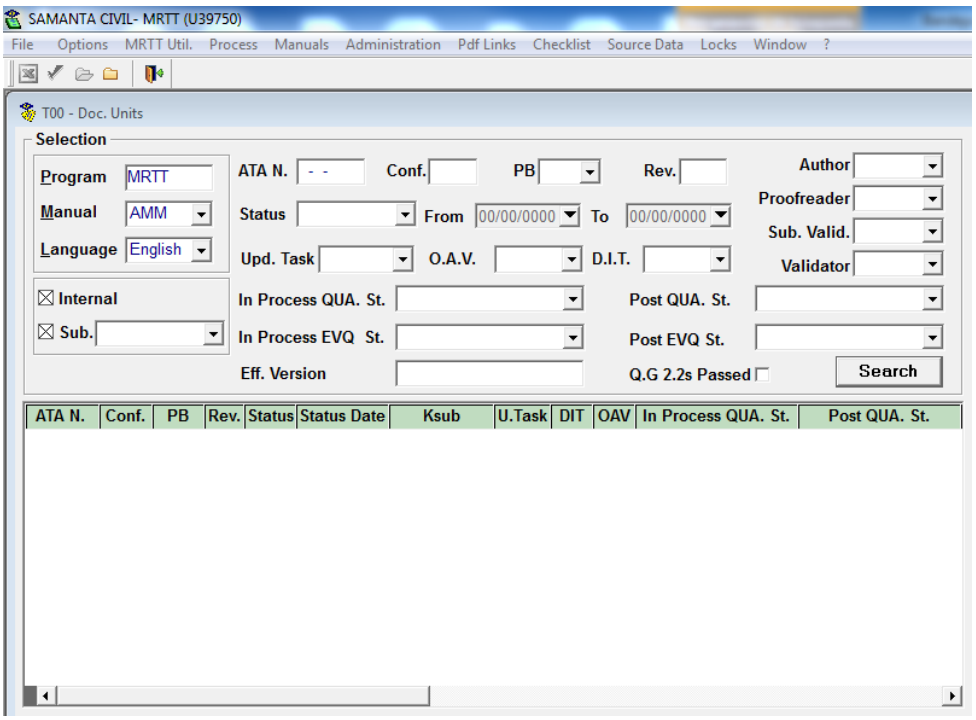


Figura 2.2: Ventana de selección de unidad a editar

2.2. Lenguajes (lenguajes de etiquetas, SGML)

A diferencia de los pequeños manuales, los manuales que se trabajan bajo la herramienta de *Samanta Civil* (además de IPC), estan compuestos por ilustraciones (que están regidas bajo una normativa propia y común para todos los manuales) y un texto. Este texto está escrito con un lenguaje de etiquetas conocido como *SGML* (Standard Generalized Markup Language).

A través de la herramienta Samanta Civil es posible *parsear* el documento que el autor técnico ha editado, pero este *parser* es limitado: comprueba si se llama dos veces a la misma ilustración desde el documento, si existe en la base de datos un *caution* o un *warning* al que se haga referencia en el documento, si se ha cometido algún error con la jerarquía de etiquetas (si se viola la estructura del SGML) o bien si se ha incumplido parte de la DTD.

La DTD (Document Type Definition) define la estructura, elementos y atributos de un documento XML [5]. Existe una DTD distinta para cada manual de los escritos con un lenguaje de marcado, aunque la mayor parte de ellos es común (emplean etiquetas y atributos a las que les dan usos muy similares).

3. Importancia de los errores

Tal y como se ha comentado, el porcentaje de errores en las PPTT no es elevado en comparación con la cantidad total de información que aportan como es lógico, pero teniendo en cuenta el volumen de información que se maneja, un pequeño porcentaje puede significar una cantidad enorme de errores. Del mismo modo que muchos de estos son insignificantes y pueden pasar desapercibidos durante años, otros sí tienen gran importancia y pueden tener consecuencias muy graves.

- **Pago a empresa subcontratada.** Cuando un cliente detecta un error en las publicaciones, abre una RFC (Request For Customer)[1] Ésta puede incurrir en gastos de producción: se debe pagar a una empresa para que edite la tarea o figura errónea de manera correcta. Este gasto no se tendría que asumir si a la hora de validar el error se hubiera detectado.
- **Sobrecarga de trabajo.** En el momento en el que se recibe el aviso o se toma la determinación de arreglar un error, la carga de trabajo aumenta de manera lógica, ya que no sólo habrá que abordar el trabajo que hay que hacer en la revisión de manera natural, sino un trabajo extra. Del mismo modo, igual que puede haber dificultades para desarrollar todo este trabajo, podrá también haber dificultades para validar todo el trabajo entregado, con la consecuencia de poder estar añadiendo nuevos errores a la publicación.
- **Pago a departamento de Publishing.** Si el cliente quiere, exige que se le arregle el error con inmediatez sin esperar a que se publique el manual en la siguiente revisión, debe lanzarse una TR (Technical Request), es decir, un "parche"sobre el manual fuera de los períodos de entrega del manual, por lo que hay que pagar a una empresa para que procese la información y la envíe al cliente.
- **Descenso en los parámetros de medición de calidad interna.** El departamento TASSD10 es el encargado de desarrollar sus propias marcas de calidad a la producción interna de las PPTT. Se tienen en cuenta varios parámetros: el que más penaliza es una RFC del cliente. Puede llegarse a recibir una llamada de atención en caso de obtener puntuaciones bajas. Con este parámetro se pueden comparar los departamentos de producción de PPTT de diferentes programas.
- **Descenso en los parámetros de medición de calidad externa.** El cliente recibe una encuesta de satisfacción donde evalúa la actuación de cada departamento envuelto en la entrega de su aeronave. Con este parámetro se pueden comparar los diversos departamentos de un programa concreto.
- **AOG (Aircraft On Ground).** En la entrega de una aeronave, si el cliente detecta información errónea o ausente de importancia, la aeronave no echará a volar hasta que se arregle. Es frecuente en la firma del contrato con el cliente acordar una cifra de descuento sobre el precio de la aeronave por cada día que una aeronave tenga que quedar en tierra por este motivo. Esta cifra suele ser muy ginificativa.

[1] Una RFC es queja, consulta o petición formal por parte de un cliente a la empresa.

4. Automatización en detección de errores

En las PPTT surge, por tanto, la necesidad de detectar estos errores de manera automática: de manera manual sería inabordable para la mayoría de casos a menos que se contase con un equipo muy grande de personas y se quisiera interrumpir la producción durante una o varias revisiones para detectar estos errores. Como esto no es lógico, en este Trabajo de Fin de Grado se trata de satisfacer esta necesidad.

El tipo de errores cuya detección podría automatizarse es enorme. En este trabajo nos vamos a centrar en varios casos y diversos, de modo que puedan sentarse las bases para la automatización del resto. Para ello, necesitaremos dos tipos de lenguajes de programación: uno que nos permita extraer la información de los documentos escritos en lenguajes de etiquetas y otro que nos permita mostrar de manera sencilla la información extraída.

Para que los resultados obtenidos sean aprovechables por todos los trabajadores del programa de las distintas empresas subcontratadas y de la misma empresa Airbus, hay que mostrarlos haciendo uso de una herramienta que todos conozcan y que todos sepan usar de manera correcta. Esta herramienta es el conocido software Microsoft Excel.

4.1. Programas y lenguajes

4.1.1. Extracción de datos

Puesto que las PPTT cuya detección de errores pretendemos automatizar están escritas con lenguaje de etiquetas (SGML) necesitamos una herramienta que nos permita desarrollar macros con un lenguaje determinado que sea eficiente para dichas búsquedas y haya sido concebido para este tipo de labores.

Llegados a este punto, debemos elegir entre los softwares que Airbus nos permite emplear en nuestros ordenadores de oficina y es preferible que tengan un bajo coste o sean gratuitos para que en un futuro cualquier trabajador pueda utilizar los códigos en un programa que no presente impedimentos para la descarga. Tenemos dos softwares que cumplen estas características:

- **BaseX**: Es uno de los softwares más conocidos para realizar extracciones similares a las que queremos acometer.Utiliza el lenguaje XQuery, que es un lenguaje de consulta especializado en colecciones de datos XML [6].

- **(First Object XML Editor)**: Es un software gratuito que emplea el lenguaje C++ y una biblioteca de funciones propia (la biblioteca *CMarkup*) para realizar las extracciones [7].

El software elegido es el FOXE por dos motivos de gran peso:

- **1.** BaseX está destinada al manejo de datos XML. Nuestros manuales están desarrollados bajo el lenguaje SGML. El lenguaje SGML es menos estricto que el XML, es más genérico. Cuando le entregamos a BaseX uno de nuestros manuales nos responde que no es un XML bien formado: esto se debe a que tenemos varias etiquetas que no tienen cierre, sólo apertura, lo que va en contra de uno de los principios que debe cumplir un archivo XML bien formado.

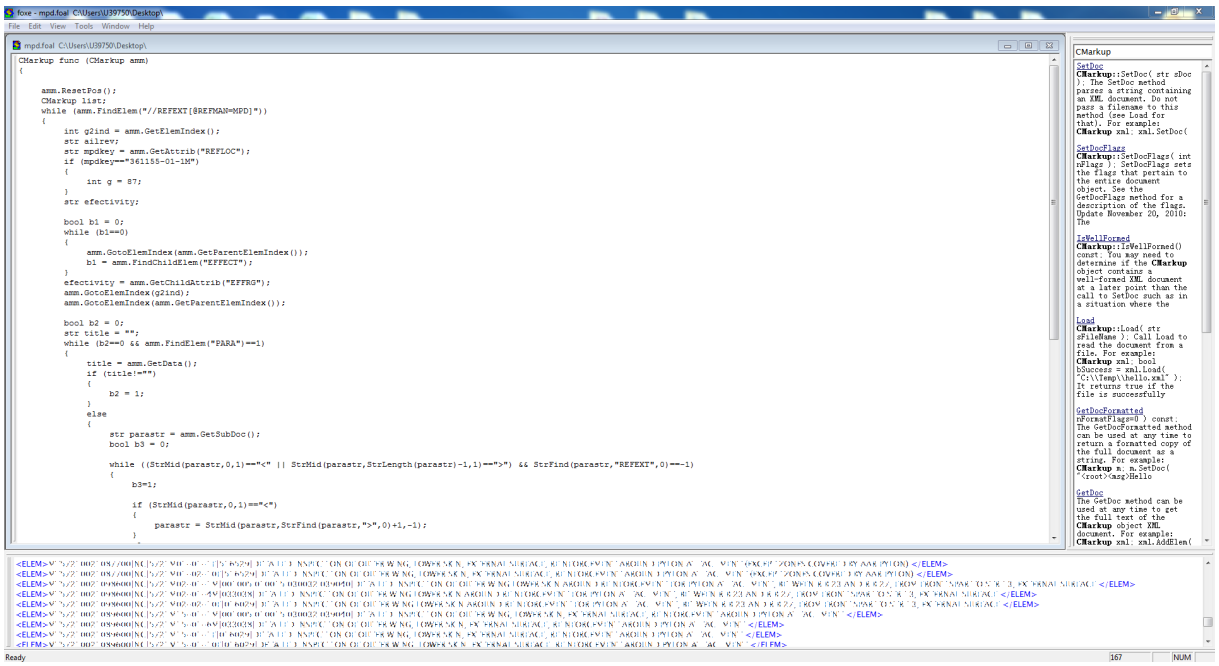


Figura 4.1: Compilador de FOXE

- **2.** BaseX no es capaz de trabajar de manera eficiente con archivos tan grandes como lo son cada uno de los manuales con los que trabajamos para las capacidades de los ordenadores de los trabajadores.

FOXE sin embargo no presenta ninguno de estos problemas. El único inconveniente de este programa es que no utiliza un lenguaje universalmente conocido como lo es el XQuery [8], por lo que es necesario un proceso de aprendizaje exclusivo para poder desarrollar macros con este software.

4.2. TSM-AMM

4.2.1. Introducción

Tal y como se ha explicado anteriormente, una entrada a las PPTT es la detección de un error en el funcionamiento de la aeronave haciendo uso del TSM. Para llevar a cabo la verificación de un error, el TSM hace uso del AMM para ejecutar determinados tests que ayuden a comprobar si el mantenimiento debe hacerse a un equipo u otro. Estas llamadas de TSM a AMM se realizan mediante hipervínculos.

En ocasiones, estos hipervínculos no se realizan correctamente: puede ser porque TSM llama a una tarea que no existe en AMM (error de Máster) o bien porque se llama a AMM desde un clientizado cuyo AMM no contiene la tarea (error de clientizado), es decir, debido a incongruencias con las efectividades de las tareas. Se hace, por tanto, necesario trazar la veracidad de los links entre estos manuales.

Tal y como se acaba de comentar, podemos clasificar un error según el tipo de publicación a la que afecte en:

- **Error de Máster.** Conocemos de esta forma a aquellos errores de link donde se llama a una tarea que ni siquiera existe en todo el manual. Se conoce con este

nombre porque los trabajadores de las PPTT tienen acceso a las publicaciones de los manuales de dos formas distintas. Al iniciar el Airn@v accederían al *MASTER*, donde aparece la información de todos los clientes. Los mismos clientes no disponen de este acceso, de modo que no pueden ver los equipos que llevan montados los aviones de otros clientes. No obstante, habitualmente se utiliza por los trabajadores de las PPTT porque te permite acceder a la información de manera más sencilla. Por tanto, si una tarea que sólo afecta al avión llama a otra que no exista en todo el manual, saltará un error.

- **Error de clientizado.** Los clientes podrán acceder únicamente a la información de sus aviones. Si existe un error de Máster, generará un error en el manual clientizado, lógicamente. Los errores de clientizado se refieren a casos en los que una tarea que aplica a un avión X del cliente llama a otra tarea del que únicamente aplica a los aviones $Y - Z$, que no pertenecen al cliente. Esto le dará un error al cliente, aunque en el *MASTER* se haría el link correctamente.

Lo que queremos extraer son todas las llamadas a AMM desde TSM que den algún tipo de error, ya sea de Máster o de clientizado, y proveer la siguiente información sobre dicho error:

- Unidad mínima con *KEY* desde la que se hace la llamada: tarea o subtarea (en TSM no tendremos pageblocks con contenido propio).
- Key de dicha unidad mínima.
- Título de la tarea en la que está la llamada (independientemente de si se encuentra en una subtarea o en una tarea).
- Atributo *AIL-REV* de la unidad mínima para poder descartar errores que no correspondan a la división de MRTT a enmendar, sino que se deberá solucionar abriendo una RFC a avión verde para que ellos analicen el caso y lo resuelvan.
- Efectividad continua de la llamada.
- Efectividad discreta de la llamada.
- Tipo de error (Máster o clientizado).

En caso de que sea un error de clientizado:

- Título de la tarea llamada de AMM.
- Efectividad continua de la tarea llamada.
- Efectividad discreta de la tarea llamada.
- Atributo *AIL-REV* de la tarea llamada.

Se empleará el software FOXE para extraer los datos de los distintos manuales y VBA para convertir los datos extraídos en información fácil de entender y tratar en Excel. De manera análoga a como se realizó en el caso de los links internos, nos apoyaremos en un archivo Excel para obtener el resultado final.

4.2.2. Paso 1: Extracción de TSM

En primer lugar, vamos a extraer todos los datos necesarios de TSM.

Para ahorrar tiempo computacional, vamos a agrupar todas las llamadas que se realicen a la misma tarea de modo que no tenga que realizarse la comprobación de un link cada vez que aparezca esa misma tarea: con una sóla vez nos basta.

Agruparemos la información, para este primer caso, en un atributo, de manera que se ahorre tiempo computacional en los programas venideros: FOXE tarda menos tiempo en buscar información dentro de atributos que dentro de textos planos.

El código empleado para esta primera extracción es el siguiente:

```
CMarkup tsm_extract( CMarkup tsm )
{
  str Path = "[insert active folder path]";
  CMarkup mList;
  tsm.ResetPos();
  tsm.FindElem("//MSTFLTAB");
  // Buscamos este elemento para posicionarnos en el manual_
  //_ y no encontrar elementos que no queremos

  int i = 1;
  while ( tsm.FindElem("//REFEXT[@REFMAN=AMM]"))
  // Buscamos en TSM llamadas a AMM
  {
    int refindex = tsm.GetElemIndex();
    str refeff;
    str refbol = "";
    str refname = tsm.GetAttrib("REFLOC");
    // Obtenemos la key del elemento al que se llama

    bool effound = 0;
    while (effound == 0)
    // Calculamos la efectividad de la llamada dentro de TSM
    {
      bool iseff = tsm.FindChildElem("EFFECT");
        if (iseff == 1)
        {
          effound = 1;
          refeff = tsm.GetChildAttrib("EFFRG");
          while (tsm.FindChildElem("SBEFF"))
          {
            refbol = refbol + tsm.GetChildAttrib("EFFRG") + " ";
          }
          refeff = refbol + refeff;
        }
        tsm.GotoElemIndex(tsm.GetParentElemIndex());
      }
      tsm.GotoElemIndex(refindex);
      bool st_found = 0;
      str subtask;
      str atribute;
```

```
      str tag;
      str title = "";
      while (st_found == 0)
      // Buscamos de qué etiqueta cuelga (Pgblk, Task o Subtask)
      {
        tag = tsm.GetTagName();
        if (tag=="SUBTASK"||tag=="TASK"||tag=="PGBLK")
        {
          st_found = 1;
          subtask = tsm.GetAttrib("KEY");
          // Key de etiqueta padre
          atribute = tsm.GetAttrib("AIL-REV");
          // Ail-rev de etiqueta padre
          int subtask_indh = tsm.GetElemIndex();
          if (tag == "SUBTASK")
          {
            tsm.GotoElemIndex(tsm.GetParentElemIndex());
            tsm.GotoElemIndex(tsm.GetParentElemIndex());
            tsm.FindChildElem("TITLE"); // Nombre etiqueta padre
            title = tsm.GetChildData();
            tsm.GotoElemIndex(subtask_indh);
          }
          else
          {
           tsm.FindChildElem("TITLE"); // Nombre etiqueta padre
            title = tsm.GetChildData();
          }
        }
        tsm.GotoElemIndex(tsm.GetParentElemIndex());
      }
      str effdisc = eff_discretizer(refeff);
      // Discretizamos efectividad
      str concatstr = tag+subtask+atribute+refname+refeff;
      str strccat2find = "//TSM[@CONCAT=" + concatstr + "]";
      bool ccatfinder = mList.FindPrevElem(strccat2find);
      // Buscamos si el caso ya se ha añadido a la lista previamente.

      if (ccatfinder == 0) // Si no se ha añadido antes, se añade
      {
        mList.ResetPos();
        str stramm2find = "//CASE[@AMMLKEY=" + refname + "]";
        bool AmmRefFinder = mList.FindElem(stramm2find);
        if (AmmRefFinder == 0)
        {
          mList.AddElem("CASE");
          mList.SetAttrib("AMMLKEY",refname);
        }
        mList.IntoElem();
        mList.AddElem("tsm");
        mList.SetAttrib("ORIGIN_TYPE",tag);
        mList.SetAttrib("ORIGIN_KEY",subtask);
        mList.SetAttrib("TASK_TITLE",title);
        mList.SetAttrib("ORIGIN_AILREV",atribute);
```

```
            mList.SetAttrib("EFF_CONT",refeff);
            mList.SetAttrib("EFF_DISC",effdisc);
            mList.SetAttrib("CONCAT",concatstr);
            // Ponemos este atributo para que sea más fácil reconocer_
            // qué casos se van a repetir si se añaden de nuevo
                mList.OutOfElem();
        }
        tsm.GotoElemIndex(refindex);
        i=i+1;
      }
      mList.ResetPos();
      while (mList.FindElem("//tsm[@CONCAT]"))
      // Eliminamos el atributo que ya no nos sirve
      {
        mList.RemoveAttrib("CONCAT");
      }
  str str2save = Path + "\\tsm_extract.xml";
  mList.Save(str2save);
  return mList;
}
```

En el manual TSM, todas las llamadas que se realizan al AMM se llevan a cabo a través de las etiquetas *REFEXT* que tienen como atributo *REFMAN* el valor *AMM*. Este código busca todas las etiquetas que complan esta condición y entonces extrae la siguiente información sobre la misma:

- Código de la llamada (tarea de AMM a la que se llama).
- Efectividad de la llamada.
- Pageblock tarea subtarea de TSM en la que se encuentra y el título del mismo.
- Atributo *AIL-REV* del pageblock tarea subtarea.

Como es posible que se llame varias veces a la misma tarea de AMM en puntos distintos del manual TSM, el resultado de nuestro programa será un documento XML formado por etiquetas $< CASE >$ que tengan como atributo *AMM_KEY* el código de la tarea llamadas. A su vez, esta etiqueta tendrá como hijas tantas etiquetas $< tsm >$ como llamadas a esa tarea haya en TSM. Cada una de estas etiquetas $< tsm >$ tendrá recogido dentro de atributos toda la información que se acaba de explicar.

El motivo por el que la información la recogeremos en atributos normalmente en vez de texto plano dentro de etiquetas, se debe a que el FOXE es capaz de hacer búsquedas mucho más rápidas de este modo.

Ponemos como ejemplo una sección del resultado obtenido:

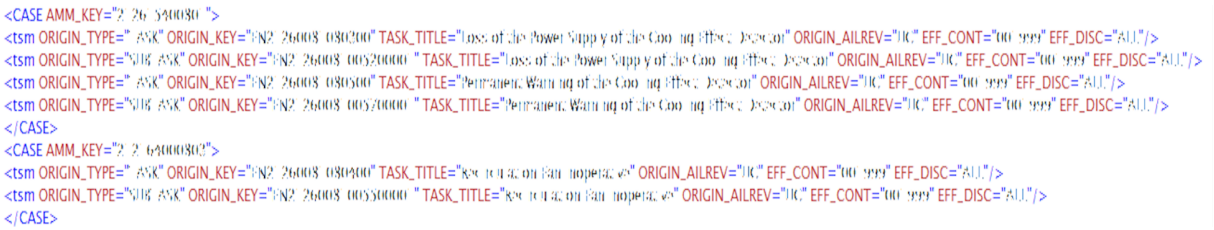


Figura 4.2: Resultado del Paso 1; Extracción de TSM

El resultado obtenido se guardará en un archivo con extensión *.xml* en la misma ruta donde se encuentran todas las macros de este programa de cruce de errores entre TSM y AMM de modo que pueda ser utilizado en el siguiente paso.

4.2.3. Paso 2: Extracción de AMM

En este caso, obtendremos los datos provenientes de AMM. Para ello, se tomará como input, además del manual AMM, los datos obtenidos del programa anterior para no tener que evaluar la información de cada tarea del manual, sino sólo las llamadas desde TSM.

Para ello empleamos el siguiente código:

```
str amm_extract( CMarkup amm )
{

str Path = "[insert active folder path]";
CMarkup mList;
CMarkup ammList;

mList.Load("C:\\Users\\U39750\\Desktop\\ErrorLinks_TSM_AMM_
_\\AmmLinksFromTsm.xml");
// Recuperamos el archivo generado generado en el programa anterior
mList.ResetPos(); // (listado de links)
int i = 1;
while (mList.FindElem("ELEM")  )
// Mientras encontremos un elemento con la etiqueta "ELEM", buscamos
{
  i=i+1;
  str ammeffect = "";
  str ammailrev;
  str ammtitle;

  str amm2an = mList.GetAttrib("at");
  int len_amm2an = StrLength(amm2an);

  if (len_amm2an == 12)
  // Buscamos así los datos de la referencia porque no se distingue_
  //entre variantes y configuraciones y hay que sumar todos los datos
  {
    amm.ResetPos();

    str str2find = "//CHAPTER[@CHAPNBR=" + StrMid(amm2an,0,2) + "]";
    bool b1 = amm.FindElem(str2find);

    str2find = "//SECTION[@SECTNBR=" + StrMid(amm2an,2,2) + "]";
    bool b2 = amm.FindChildElem(str2find);

    amm.GotoElemIndex(amm.GetChildElemIndex());
    str2find = "//SUBJECT[@SUBJNBR=" + StrMid(amm2an,4,2) + "]";
    bool b3 = amm.FindChildElem(str2find);

    bool b4 = 0;
    amm.GotoElemIndex(amm.GetChildElemIndex());
```

```
    while (amm.FindChildElem("PGBLK"))
    {
      int pgblk_ind = amm.GetChildElemIndex();
      amm.GotoElemIndex(pgblk_ind);
      while (amm.FindChildElem("TASK"))
      {
      if (amm.GetChildAttrib("FUNC")==StrMid(amm2an,6,3) _\\
      _ amm.GetChildAttrib("SEQ")==StrMid(amm2an,9,3))
      {
        int task_index = amm.GetChildElemIndex();
        amm.GotoElemIndex(task_index);
        amm.FindChildElem("EFFECT");
        amm.GotoElemIndex(amm.GetChildElemIndex());
        while (amm.FindChildElem("SBEFF"))
        {
          ammeffect = ammeffect + amm.GetChildAttrib("EFFRG") + " ";
        }
        ammeffect = ammeffect + amm.GetAttrib("EFFRG");

        amm.GotoElemIndex(task_index);
        ammtitle = amm.FindChildElem("TITLE");
        ammtitle = amm.GetChildData();
        str new_ammailrev = amm.GetAttrib("AIL-REV");

        if (ammailrev == "UC" && (new_ammailrev == "RC" _
        _|| new_ammailrev == "NC"))
        {
          ammailrev = new_ammailrev;
        }
        else if (ammailrev == "RC" && new_ammailrev == "NC")
        {
          ammailrev = new_ammailrev;
        }
        else if (ammailrev == "")
        {
          ammailrev = new_ammailrev;
        }

        amm.GotoChildElemIndex(task_index);
        b4 = 1;
      }
    }
    amm.GotoChildElemIndex(pgblk_ind);
  }

  if (b1*b2*b3*b4==1)
  // Si se ha encontrado al menos una referencia, el link_
  //_existirá al menos en el Máster
  {
    ammList.AddElem("CASE");
    // Se meten todos los datos como atributos para facilitar_
    //_posteriores búsquedas
    ammList.SetAttrib("REF",amm2an);
```

```
    ammList.SetAttrib("TITLE",ammtitle);
    ammList.SetAttrib("EFFECT",ammeffect);
    ammList.SetAttrib("AIL-REV",ammailrev);
   }
  }
}
str str2save = Path + "\\amm_extract.xml";
mList.Save(str2save);

return ammList;
}
```

Buscando dentro de todas las etiquetas $< CASE >$ el atributo *AMM_KEY* en el resultado obtenido en el paso anterior obtenemos todas las tareas de AMM de las que necesitamos extraer información. Cada una de estas *keys* se buscan en AMM. Si no se encuentran, entonces tendremos directamente un error de Máster tal y como se explicó. En caso de encontrar la tarea, obtenemos la siguiente información:

- Título del pageblock tarea subtarea.
- Atributo *AIL-REV* del mismo.
- Efectividad

Se generará nuevamente un archivo con extensión *.xml* que recoja bajo la etiqueta $< CASE >$ todos estos datos con un atributo para cada uno de ellos. De no haber clasificado la información como se hizo en el paso anterior, se habría extraído varias veces la misma información de AMM, requiriendo un mayor esfuerzo computacional. Para las tareas no encontradas, no se añade absolutamente nada.

Sección ilustrativa del resultado obtenido:

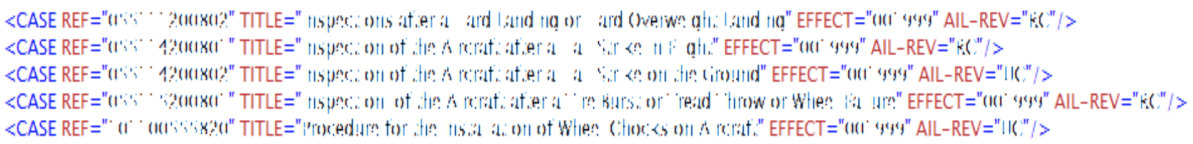


Figura 4.3: Resultado del Paso 2; Extracción de AMM

4.2.4. Paso 3: Cruzamos los datos extraídos

Una vez obtenidos los datos de AMM y TSM afectados por las llamadas entre estos manuales, es necesario corroborar la correcta correspondencia entre dichas llamadas. Usamos de nuevo el lenguaje *CMarkup*. Nuestro código queda así:

```
CMarkup ERROR_TRACER (CMarkup mDocToQuery)
{

str Path = "[insert active folder path]";
CMarkup tsmList;
CMarkup ammList;
CMarkup eList;

str str2tsm = Path + "\\tsm_extract.xml";
```

```
str str2amm = Path + "\\amm_extract.xml";

tsmList.Load(str2tsm);
// Cargamos datos de Program1.foal (datos TSM)
ammList.Load(str2amm);
// Cargamos datos de Program2.foal (datos AMM)

tsmList.ResetPos();

while (tsmList.FindElem("//CASE"))
// Analizamos cada caso del listado de llamadas desde TSM
{
  str key2an = tsmList.GetAttrib("AMMKEY");
  // Obtenemos del listado de TSM la key que queremos _
  //_encontrar en el listado de AMM

  ammList.ResetPos();
  str s2LfInAmm = "//CASE[@REF=" + key2an + "]";
  if (ammList.FindElem(s2LfInAmm))
  {
    CMarkup TSM_Pack = "";
   bool checker = 0;

   while (tsmList.FindChildElem("tsm"))
   // Analizamos cada llamada de TSM al pgblk/tarea_
   //_/subtarea en cuestión
    {
      str wrong_eff = "";
      // Comparamos que cada MSN (obtenido de la efect. discreta de_
      //TSM) se encuentre en la ef. discreta de AMM (y NO al revés)
          str tsm_eff = tsmList.GetChildAttrib("EFF_DISC");
          if (tsm_eff == "ALL")
          {
            tsm_eff = "/1//2//3/[...]/60/";
          }
          str amm_eff = ammList.GetAttrib("EFF_DISC");

          int tsmlen = StrLength(tsm_eff);
          int j = 0;

          if (amm_eff!="ALL")
          {
            while (j<tsmlen)
                {
                  str plane2an = StrMid(tsm_eff,j,StrFind_
                  _(tsm_eff,"/",j+1)-j+1);
                   if (StrFind(amm_eff,plane2an,0) == -1)
                   {
                        if (StrLength(plane2an)==5)
                        {
                           str plane2an_v2 = "/" + StrMid_
                           _(plane2an,2,2) + "/";
                           if (StrFind(amm_eff,plane2an_v2,0) == -1)
```

```
                        {
                                wrong_eff = wrong_eff + plane2an;
                            }
                    }
                    else
                    {
                      wrong_eff = wrong_eff + plane2an;
                    }
                }
                j=StrFind(tsm_eff,"/",j+1)+1;
            }
        }
        if (wrong_eff != "")
        // Si tenemos alguna efectividad errónea, entonces _
        //_generamos la unidad de fallo CMarkup de TSM
        {
          checker = 1;

              CMarkup FromTsm2Add;
              FromTsm2Add.AddElem("tsm");
              FromTsm2Add.SetAttrib("ORIGIN_TYPE",tsmList._
              _GetChildAttrib("ORIGIN_TYPE"));

              FromTsm2Add.SetAttrib("ORIGIN_KEY",tsmList._
              _GetChildAttrib("ORIGIN_KEY"));

              FromTsm2Add.SetAttrib("TASK_TITLE",tsmList._
              _GetChildAttrib("TASK_TITLE"));

              FromTsm2Add.SetAttrib("ORIGIN_AILREV",tsmList._
              _GetChildAttrib("ORIGIN_AILREV"));

              FromTsm2Add.SetAttrib("EFF_CONT",tsmList._
              _GetChildAttrib("EFF_CONT"));

              FromTsm2Add.SetAttrib("EFF_DISC",tsmList._
              _GetChildAttrib("EFF_DISC"));

              FromTsm2Add.SetAttrib("FAILING_EFF_DISC",wrong_eff);
              TSM_Pack = TSM_Pack + FromTsm2Add;
        }
      }

    if (checker == 1)
    // Si hay fallo, genermaos la unidad de fallo de AMM _
    //_y apuntamos el caso en la lista de errores final
    {
      CMarkup AMM_Pack;
      AMM_Pack.AddElem("amm");
      AMM_Pack.SetAttrib("TITLE",ammList.GetAttrib("TITLE"));
      AMM_Pack.SetAttrib("EFF_CONT",ammList._
      _GetAttrib("EFF_CONT"));
```

```
            AMM_Pack.SetAttrib("EFF_DISC",ammList._
            _GetAttrib("EFF_DISC"));

            AMM_Pack.SetAttrib("AIL-REV",ammList.GetAttrib("AIL-REV"));

            CMarkup Full_Pack;
            Full_Pack.AddElem("ERROR");
            Full_Pack.SetAttrib("ERROR_TYPE","CUSTOMIZED");
            Full_Pack.SetAttrib("WRONG_LINK",ammList.GetAttrib("REF"));
            Full_Pack.IntoElem();
            Full_Pack.AddElem("TSM");
            Full_Pack.SetElemContent(TSM_Pack);
            Full_Pack.AddElem("AMM");
            Full_Pack.SetElemContent(AMM_Pack);
            Full_Pack.OutOfElem();
            Full_Pack.OutOfElem();

            eList = eList + Full_Pack;
        }
    }
    else
    // Si el link no se ha encontrado en el listado de AMM, _
    //_entonces hay un error. Se marca como tal
    {
        eList.AddElem("ERROR");
            eList.SetAttrib("CALL_TASK",key2an);
            eList.SetAttrib("ERROR_TYPE","MASTER");
            eList.IntoElem();
            eList.AddElem("TSM");
            eList.SetElemContent(tsmList.GetElemContent());
            CMarkup FromTsm2Add = tsmList.GetElemContent();
            eList.SetElemContent(FromTsm2Add);
            eList.OutOfElem();
    }
}

eList.ResetPos();

while (eList.FindElem("//ERROR"))
// Una vez generado el CMarkup de errores, le añadimos_
//_ datos identificativos a la cabecera de cada error
{
    str errorkind = eList.GetAttrib("ERROR_TYPE");

    int error_index = eList.GetElemIndex();
    str ErrorChain = "";

    eList.FindChildElem("TSM");
    eList.GotoElemIndex(eList.GetChildElemIndex());

    while (eList.FindChildElem("tsm"))
    // Calculamos para cada link de AMM cuántos fallos (en MSNs)_
    //_ hay en total teniendo en cuenta todas las llamadas desde TSM
```

```
  {
        str ef2disc="";
        if (errorkind == "CUSTOMIZED")
        // Error de customizado significa que el link existe pero_
        //la ef. en AMM no comprende todos los aviones de TSM.
        {
           ef2disc = eList.GetChildAttrib("FAILING_EFF_DISC");
        }
        else
        {
          ef2disc = eList.GetChildAttrib("EFF_DISC");
          // Metemos todas las efectividades de las tareas _
          //_afectadas aunque se repitan efectividades
         }

         if (ef2disc == "ALL")
         {
               ef2disc = "/1//2//3/[...]/60/";
         }

         int elen = StrLength(ef2disc);

         int i = 0;
         while (i<elen)
         // Depuramos efectividades repetidas
         {
               int wherebar = StrFind(ef2disc,"/",i+1);
               str pl2an = StrMid(ef2disc,i,wherebar-i+1);
               i = wherebar+1;

               if (StrFind(ErrorChain,pl2an,0)==-1)
               {
                     ErrorChain = ErrorChain + pl2an;
               }
         }
   }
   eList.GotoElemIndex(error_index);

        if (ErrorChain == "/1//2//3/[...]/60/")
        {
               ErrorChain = "ALL";
        }

        eList.SetAttrib("WRONG_EF",ErrorChain);
  }
  str str2save = Path + "\\Wrong_Links.xml";
  mList.Save(str2save);

  return eList;
}
```

El objetivo de este código es tomar los dos archivos con extensión *.xml* generados en los pasos anteriores y para cada caso de TSM, cruzar los datos con los obtenidos en AMM, es decir, para cada caso $< tsm >$ del listado de TSM comprobar si está en algún

$< CASE >$ de AMM: si no está sería error de Máster y si no todos los aviones para los que tiene efectividad en el listado de TSM están incluidos en la efectividad del listado de AMM, entonces es un error de clientizado.

El resultado será un XML que contenga sólo los casos de error. Cada caso estará recogido bajo la etiqueta $< CASE >$, teniendo como atributos la KEY del elemento llamado de AMM y las efectividades erróneas en caso de ser un error de clientizado. Además tendrá un atributo que clasifique de entrada qué tipo de error es. En caso de ser un error de Máster, contendrá únicamente la información de TSM extraída en el Paso 1, pero, en caso contrario, tendrá el conjunto de información extraída en los Pasos 1 y 2.

Una sección del resultado obtenido puede verse en la Figura 4.4.

Este archivo se guardará en la carpeta donde estamos trabajando a modo de *backup*. No obstante, para el siguiente paso necesitamos convertir los datosde manera más fácil para tratar con VBA. Usamos el siguiente código para organizar los datos en forma de filas cuya información queda separada con el caracter "|". Para ello, deberíamos sustituir la última línea del código anterior por lo siguiente:

```
CMarkup finalList;
eList.ResetPos();
while (eList.FindElem("ERROR"))
{
  str tsmdata;
  str ammdata;
  str eff_Fail;
  if (eList.GetAttrib("ERROR_TYPE")=="CUSTOMIZED")
  {
    int eind = eList.GetElemIndex();
    eList.FindChildElem("//amm");
    ammdata = eList.GetChildAttrib("TITLE") + "|" + _
    _eList.GetChildAttrib("EFF_CONT") + "|" + _
    _eList.GetChildAttrib("EFF_DISC") + "|" +_
    _eList.GetChildAttrib("AIL-REV");

    eList.GotoElemIndex(eind);
    eList.FindChildElem("TSM");
    eList.GotoElemIndex(eList.GetChildElemIndex());
    while(eList.FindChildElem("tsm"))
    {
      tsmdata = eList.GetChildAttrib("ORIGIN_TYPE") + "|" + _
      _eList.GetChildAttrib("ORIGIN_KEY") + "|" + _
```


<ERROR CALL_TASK="[illegible]" ERROR_TYPE="MASTER" WRONG_EF="ALL">
<TSM>
<tsm ORIGIN_TYPE="TASK" ORIGIN_KEY="[illegible]" TASK_TITLE="[illegible]" ORIGIN_AILREV="UC" EFF_CONT="001999" EFF_DISC="ALL"/>
<tsm ORIGIN_TYPE="SUBTASK" ORIGIN_KEY="[illegible]" TASK_TITLE="[illegible]" ORIGIN_AILREV="UC" EFF_CONT="001999" EFF_DISC="ALL"/>
<tsm ORIGIN_TYPE="TASK" ORIGIN_KEY="[illegible]" TASK_TITLE="[illegible]" ORIGIN_AILREV="UC" EFF_CONT="010015 030032" EFF_DISC="/10//11//12//13//14//15//30//31//32/"/>
<tsm ORIGIN_TYPE="SUBTASK" ORIGIN_KEY="[illegible]" TASK_TITLE="[illegible]" ORIGIN_AILREV="UC" EFF_CONT="010015 030032" EFF_DISC="/10//11//12//13//14//15//30//31//32/"/>
</TSM>
</ERROR>
<ERROR ERROR_TYPE="CUSTOMIZED" WRONG_LINK="[illegible]" WRONG_EF="/39//40/">
<TSM><tsm ORIGIN_TYPE="TASK" ORIGIN_KEY="[illegible]" TASK_TITLE="[illegible]" ORIGIN_AILREV="UC" EFF_CONT="039040" EFF_DISC="/39//40/" FAILING_EFF_DISC="/39//40/"/>
<tsm ORIGIN_TYPE="SUBTASK" ORIGIN_KEY="[illegible]" TASK_TITLE="[illegible]" ORIGIN_AILREV="UC" EFF_CONT="039040" EFF_DISC="/39//40/" FAILING_EFF_DISC="/39//40/"/>
</TSM>
<AMM><amm TITLE="[illegible]" EFF_CONT="999999" EFF_DISC="NULL" AIL-REV="RC"/>
</AMM>
</ERROR>

Figura 4.4: Resultado del Paso 3; Cruce de datos

```
      _eList.GetChildAttrib("TASK_TITLE") + "|" + _
      _eList.GetChildAttrib("ORIGIN_AILREV") + "|" + _
      _eList.GetChildAttrib("EFF_CONT") + "|" + _
      _eList.GetChildAttrib("EFF_DISC");

      eff_Fail = eList.GetChildAttrib("FAILING_EFF_DISC");
      int tsmind1 = eList.GetChildElemIndex();
      eList.GotoElemIndex(eind);
      finalList.AddElem("ELEM",eList.GetAttrib("WRONG_LINK") + "|" +_
      _ tsmdata + "|" + ammdata + "|" + eff_Fail);

      eList.GotoChildElemIndex(tsmind1);
    }
    eList.GotoElemIndex(eind);
  }
  else
  {
    int eind = eList.GetElemIndex();
    ammdata = "-" + "|" + "-" + "|" + "-" + "|" + "-";
    eList.FindChildElem("TSM");
    eList.GotoElemIndex(eList.GetChildElemIndex());
    while(eList.FindChildElem("tsm"))
    {
    tsmdata = eList.GetChildAttrib("ORIGIN_TYPE") + "|" + _
    _eList.GetChildAttrib("ORIGIN_KEY") + "|" + _
    _eList.GetChildAttrib("TASK_TITLE") + "|" + _
    _eList.GetChildAttrib("ORIGIN_AILREV") + "|" + _
    _eList.GetChildAttrib("EFF_CONT") + "|" + _
    _eList.GetChildAttrib("EFF_DISC");

    eff_Fail = eList.GetChildAttrib("FAILING_EFF_DISC");
    int tsmind1 = eList.GetChildElemIndex();
    eList.GotoElemIndex(eind);
    finalList.AddElem("ELEM",eList.GetAttrib("CALL_TASK") + "|" + _
    _tsmdata + "|" + ammdata + "|" + eff_Fail);

    eList.GotoChildElemIndex(tsmind1);
    eList.GotoChildElemIndex(tsmind1);
    }
    eList.GotoElemIndex(eind);
  }
}
return finalList;
```

4.2.5. Paso 4: Presentación de datos

Copiando el resultado obtenido en el último paso, nos vamos al archivo Excel de macros asociado a la carpeta en la que estamos trabajando, pegamos en la primera columna los datos obtenidos y pulsamos el botón " EJECUTAR ".

Podemos ver dos pestañas: una con los datos para ser finalmente estudiados y otra pestaña que muestra las estadísticas de los resultados obtenidos.

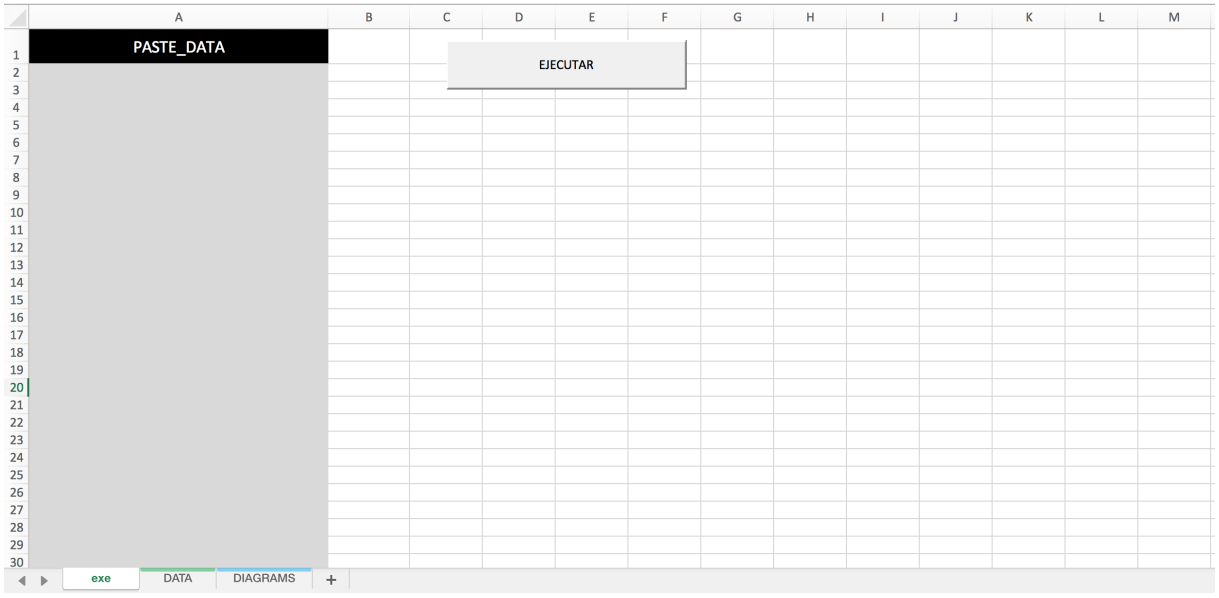


Figura 4.5: Ventana de ejecución de macro

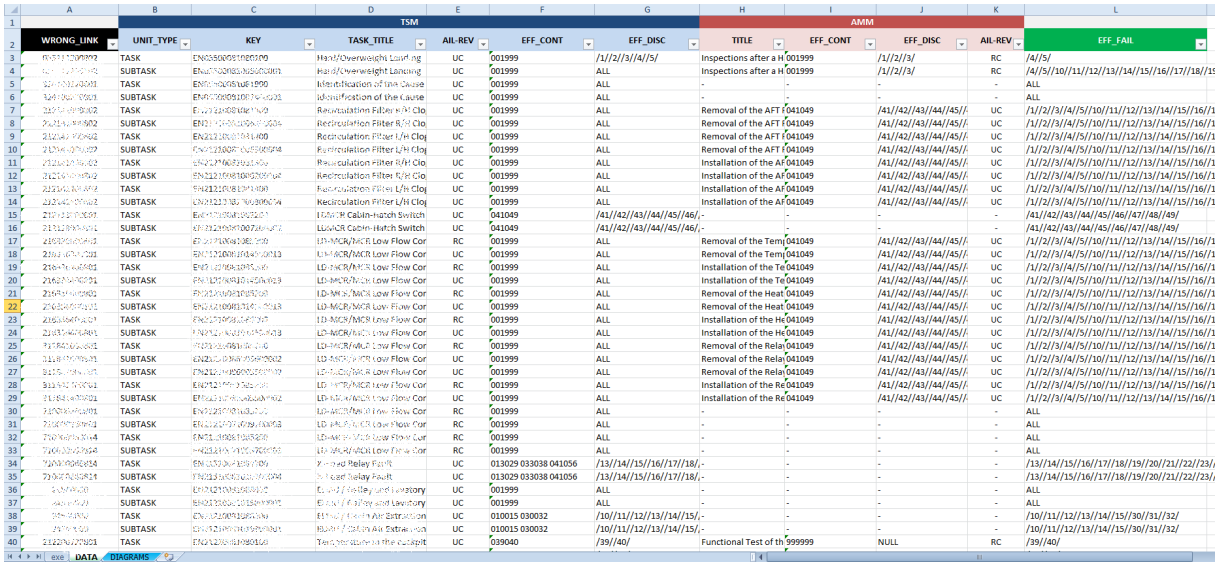


Figura 4.6: Ventana de resultados

El sencillo código de VBA que transforma los datos del Paso 3 en las pestañas de resultados es el siguiente:

```
Sub data_converter ()

Cells(2, 1).Select
Dim i As Integer

'Eliminamos etiquetas
While ActiveCell.Value <> ""
Dim fulstr As String
fulstr = ActiveCell.Value
fulstr = Mid(fulstr, InStr(fulstr, ">") + 1)
fulstr = Mid(fulstr, 1, InStr(fulstr, "<") - 1)

'Separamos en columnas
i = 1
While InStr(fulstr, "|") <> 0
```

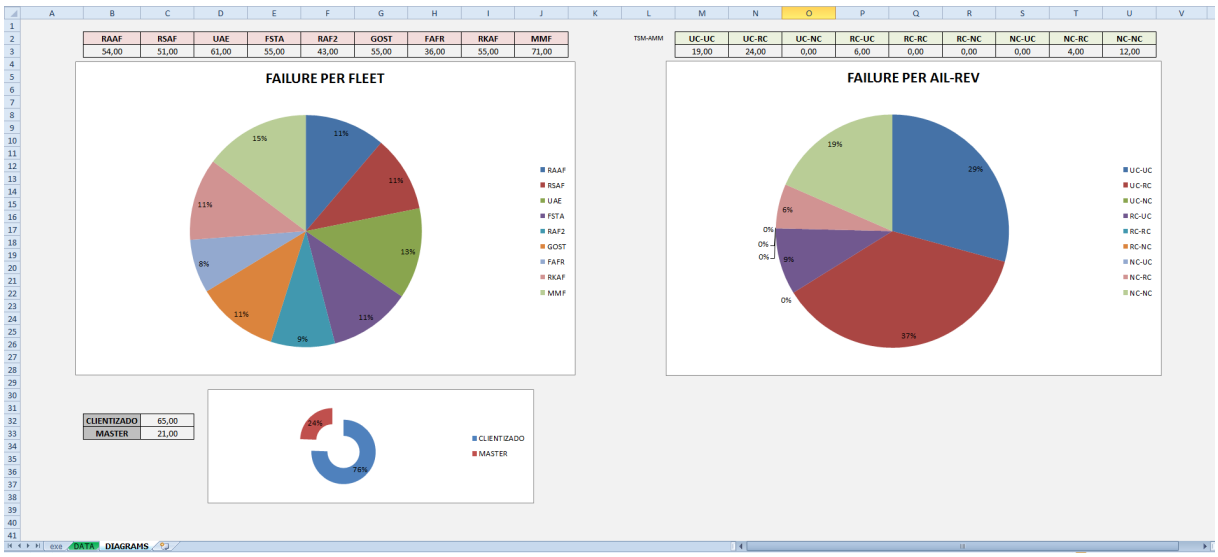


Figura 4.7: Ventana de estadísticas de resultados

```
ActiveCell.Offset(0,i).Value = Mid(fulstr, 1, InStr(fulstr, "|") - 1)
fulstr = Mid(fulstr, InStr(fulstr, "|") + 1)
i = i + 1
Wend
ActiveCell.Offset(0, i).Value = fulstr
ActiveCell.Offset(1, 0).Select
Wend

Dim j As Integer
j = Cells(Rows.Count, 1).End(xlUp).Row

'Copiamos todos los datos
Range("B2", "M" & j).Copy

'Los pegamos en la pestaña DATA
Sheets("DATA").Select
Cells(3, 1).Select
ActiveCell.PasteSpecial

'Si es error de Máster, EFF_FAIL = EFF_DISC
Cells(3, 11).Select
While ActiveCell.Value <> ""
If ActiveCell.Value = "-" Then
ActiveCell.Offset(0, 1).Value = ActiveCell.Offset(0, -4).Value
End If
ActiveCell.Offset(1, 0).Select
Wend

'Eliminamos datos no válidos
Sheets("exe").Select
Range("B2", "M" & j).Value = ""

'Calculamos datos para los gráficos
Sheets("DATA").Select
Cells(3, 12).Select
Dim RAAF As Integer
Dim RSAF As Integer
Dim UAE As Integer
Dim FSTA As Integer
```

```
Dim RAF2 As Integer
Dim GOST As Integer
Dim FAFR As Integer
Dim RKAF As Integer
Dim MMF As Integer

Dim ucuc As Integer
Dim ucrc As Integer
Dim ucnc As Integer
Dim rcuc As Integer
Dim rcrc As Integer
Dim rcnc As Integer
Dim ncuc As Integer
Dim ncrc As Integer
Dim ncnc As Integer

Dim master As Integer
Dim clientizado As Integer

While (ActiveCell.Value <> "")
'Relativo al primer gráfico
If (ActiveCell.Value = "ALL") Then
RAAF = RAAF + 1
RSAF = RSAF + 1
UAE = UAE + 1
FSTA = FSTA + 1
RAF2 = RAF2 + 1
GOST = GOST + 1
FAFR = FAFR + 1
RKAF = RKAF + 1
MMF = MMF + 1
Else
If (InStr(ActiveCell,"/1/") <> 0 [...] _
_Or InStr(ActiveCell,"/5/") <> 0) Then
'Condición para todos los MSN del 1 al 5
RAAF = RAAF + 1
End If

If (InStr(ActiveCell, "/10/") <> 0 [...]_
_Or InStr(ActiveCell, "/32/") <> 0 )Then
'Condición para todos los MSN del 10 al 12 y 30 al 32
RSAF = RSAF + 1
End If

If (InStr(ActiveCell, "/13/") <> 0 [...]_
_ Or InStr(ActiveCell, "/15/") <> 0) Then
'Condición para todos los MSN del 13 al 15
UAE = UAE + 1
End If

If (InStr(ActiveCell, "/16/") <> 0 [...]_
_ Or InStr(ActiveCell, "/18/") <> 0 Then
'Condición para todos los MSN del 16 al 29, 516 al 529 y 616 al 629
FSTA = FSTA + 1
End If

If (InStr(ActiveCell, "/39/") <> 0 Or_
_ InStr(ActiveCell, "/40/") <> 0) Then
RAF2 = RAF2 + 1
End If
```

```
If (InStr(ActiveCell, "/33/") <> 0 [...] _
_Or InStr(ActiveCell, "/38/") <> 0) Then
'Condición para todos los MSN del 33 al 38
GOST = GOST + 1
End If

If (InStr(ActiveCell, "/41/") <> 0 [...] _
_Or InStr(ActiveCell, "/49/") <> 0) Then
'Condición para todos los MSN del 41 al 49
FAFR = FAFR + 1
End If

If (InStr(ActiveCell, "/50/") <> 0 [...] _
_Or InStr(ActiveCell, "/53/") <> 0) Then
'Condición para todos los MSN del 50 al 53
RKAF = RKAF + 1
End If

If (InStr(ActiveCell, "/54/") <> 0 [...] _
_Or InStr(ActiveCell, "/60/") <> 0) Then
'Condición para todos los MSN del 54 al 60
MMF = MMF + 1
End If
End If

'Relativo al segundo gráfico
If ActiveCell.Offset(0, -7).Value = "UC" And [...] _
_ActiveCell.Offset(0, -1).Value = "UC" Then
ucuc = ucuc + 1
ElseIf ActiveCell.Offset(0, -7).Value = "UC" And [...] _
_ActiveCell.Offset(0, -1).Value = "RC" Then
ucrc = ucrc + 1
ElseIf ActiveCell.Offset(0, -7).Value = "UC" And [...] _
_ActiveCell.Offset(0, -1).Value = "NC" Then
ucnc = ucnc + 1
ElseIf ActiveCell.Offset(0, -7).Value = "RC" And [...] _
_ActiveCell.Offset(0, -1).Value = "UC" Then
rcuc = rcuc + 1
ElseIf ActiveCell.Offset(0, -7).Value = "RC" And [...] _
_ActiveCell.Offset(0, -1).Value = "RC" Then
rcrc = rcrc + 1
ElseIf ActiveCell.Offset(0, -7).Value = "RC" And [...] _
_ActiveCell.Offset(0, -1).Value = "NC" Then
rcnc = rcnc + 1
ElseIf ActiveCell.Offset(0, -7).Value = "NC" And [...] _
_ActiveCell.Offset(0, -1).Value = "UC" Then
ncuc = ncuc + 1
ElseIf ActiveCell.Offset(0, -7).Value = "NC" And [...] _
ActiveCell.Offset(0, -1).Value = "RC" Then
ncrc = ncrc + 1
ElseIf ActiveCell.Offset(0, -7).Value = "NC" And [...] _
_ActiveCell.Offset(0, -1).Value = "NC" Then
ncnc = ncnc + 1
End If

'Relativo al tercer gráfico
If ActiveCell.Offset(0, -1) = "-" Then
master = master + 1
Else
```

```
clientizado = clientizado + 1
End If

ActiveCell.Offset(1, 0).Select
Wend

'Pegamos los datos en la hoja de gráficos
Sheets("DIAGRAMS").Select

'Relativo al primer gráfico
Cells(3, 2).Value = RAAF
Cells(3, 3).Value = RSAF
Cells(3, 4).Value = UAE
Cells(3, 5).Value = FSTA
Cells(3, 6).Value = RAF2
Cells(3, 7).Value = GOST
Cells(3, 8).Value = FAFR
Cells(3, 9).Value = RKAF
Cells(3, 10).Value = MMF

'Relativo al segundo gráfico
Cells(3, 13).Value = ucuc
Cells(3, 14).Value = ucrc
Cells(3, 15).Value = ucnc
Cells(3, 16).Value = rcuc
Cells(3, 17).Value = rcrc
Cells(3, 18).Value = rcnc
Cells(3, 19).Value = ncuc
Cells(3, 20).Value = ncrc
Cells(3, 21).Value = ncnc

'Relativo al tercer gráfico
Cells(32, 3).Value = clientizado
Cells(33, 3).Value = master

End Sub
```

4.3. Links internos AMM

4.3.1. Introducción

A la hora de realizar una labor de mantenimiento sobre un equipo determinado, hay subtareas que comparte de manera idéntica con otras tareas de mantenimiento de otros equipos. Con el objetivo de no duplicar la información del manual y hacerlo a su vez más pesado, se separan estas subtareas para formar tareas independientes de modo que en un punto determinado, otra tarea puede llamarla para que se ejecuten concretamente esos pasos y posteriormente seguir describiendo el resto de la tarea. Es similar al comportamiento de una función de un código de programación: por sí misma no tiene utilidad, pero sirve a otras funciones para procesar unidades determinadas de datos o variables.

Con el objetivo de hacer del uso de los manuales una labor dinámica, las llamadas a otras tareas se realizan mediante hipervínculos. De este modo, para incluir la llamada a otra tarea del manual dentro de una, debe hacerse de acuerdo a la DTD, de modo que el motor de búsqueda del software encargado de la publicación para el cliente (Airn@v) pueda detectar correctamente la información sobre la tarea que debe buscar y abrir.

En ocasiones, estos datos no se introducen de manera correcta, de modo que cuando un cliente clica sobre un hipervínculo, éste no le conduce a ninguna tarea o bien le lleva a una tarea incorrecta. En otros casos, aunque se introduce bien la referencia a otra tarea, en revisiones posteriores se elimina o modifica aquella tarea y no se actualiza la tarea que la llamaba, de modo que se genera un nuevo error.

La extracción que hagamos debe presentar todos los fallos posibles, por lo que no sólo compararemos que las llamadas a tareas se correspondan con tareas reales, sino que todos los aviones con aplicabilidad para la llamada a la tarea estén contenidos en la efectividad de dicha tarea. Si D es el rango de aviones con efectividad para la llamada a la tarea y C es el rango de la tarea a la que se llama, debemos comprobar que $D \in C$.

Debemos proporcionar el atributo *AIL-REV* de las llamadas y de las tareas llamadas, para que al analizar los datos puedan cribarse aquellos hipervínculos cuyo fallo no recae sobre la división militar de la empresa, sino que deben reportarse los errores a Airbus Civil para que ellos los solucionen.

Será necesario también, además de la ubicación de la llamada, que puede ubicarse en una subtarea, tarea o pageblock, la tarea a la que pertence la subtarea en caso de que ésta sea la ubicación de dicha llamada. No nos arrojaría ninguna información útil conocer el atributo *AIL-REV* de la tarea ni su efectividad: únicamente nos interesan estos datos de la llamada. Puesto que las empresas subcontratadas cobran por *Data Modules*, que por definición son las unidades mínimas de trabajo, al responsable de un manual el dato numérico que más le interesa es el número de tareas que deben editar. Podemos considerar los pageblocks 001 (descriptivos) como tareas: son pageblocks que no contienen tareas en su interior y se cobran como si fueran una tarea.

Vamos a dividir este proceso en 4 pasos con el objetivo de reducir poderosamente el tiempo computacional de la extracción. Al comienzo de la descripción del segundo paso se explica por qué.

4.3.2. Paso 1: Detección de hipervínculos en AMM

Debemos desarrollar una macro capaz de extraer en primera instancia:

- **DAD TASK**. En caso de que la llamada provenga de una tarea o un pageblock 001, se corresponderá con la misma tarea o pageblock. En caso de ser una tarea, deberá proporcionarnos su tarea padre.
- **DAD KEY**. Es la unidad documental padre, entiendo como menor padre posible la menor unidad que contiene su propia KEY como elemento diferenciador único, es decir, una subtarea. Debe proporcionarse dicha KEY para localizar de manera precisa dónde se encuentra la llamada.
- **DAD AILREV**. Proporciona el atributo de programa de edición de la unidad mínima.
- **REFINT KEY**. Contiene el código de llamada a una tarea. Será el atributo "$REFID$" de la etiqueta $< REFINT >$ en la que se encuentre la llamada.
- **REFINT EFF**. La efectividad de la llamada. Esta efectividad se dará de manera continua en este primer paso tal y como aparece en el manual.

Como hablamos de una extracción de datos, emplearemos el software FOXE. El código con el que obtendremos estos resultados será el siguiente:

```
CMarkup First_Func(CMarkup amm)
{
  str Path = "[insert active folder path]";
  amm.ResetPos();
  CMarkup wrList;

  while (amm.FindElem("//REFINT"))
  // Buscamos todas las etiquetas REFINT de AMM
  {
    int ActRefInt = amm.GetElemIndex();
    str rec_dad;
    str EffRef;
    str DadKey;
    str AilRev;
    str RefKey = amm.GetAttrib("REFID");
    // El atributo REFID contiene la key del elemento a analizar

    bool eff_b = 1;
    // Si vale 1 si tenemos que seguir buscando la efectividad
    bool dad_b = 1;
    // Si vale 1 si tenemos que seguir buscando al padre

    while (eff_b == 1||dad_b == 1)
    {
      amm.GotoElemIndex(amm.GetParentElemIndex());
      int TempInd = amm.GetElemIndex();
      str TagName = amm.GetTagName();

      if (TagName=="PGBLK"||TagName=="TASK"||TagName=="SUBTASK")
      {
        DadKey = amm.GetAttrib("KEY");        // Key del padre
        AilRev = amm.GetAttrib("AIL-REV");    // Atributo del padre
        dad_b = 0;

        if (amm.GetTagName() == "SUBTASK")
        {
          int returning_index = amm.GetElemIndex();
          amm.GotoElemIndex(amm.GetParentElemIndex());
          amm.GotoElemIndex(amm.GetParentElemIndex());
          rec_dad = amm.GetAttrib("KEY");
          amm.GotoElemIndex(returning_index);
        }
        else
        {
          rec_dad = DadKey;
        }
      }
      if (eff_b == 1)
      {
      // La efectividad la hereda del propio REFINT si tiene etiqueta_
      //_EFFECT y si no, del primer padre con hijo EFFECT.
      // Si pertenece a un REFLOC le asociamos la efectividad del REFLOC
        if (amm.GetTagName()=="REFBLOCK")
        {
          RefKey = "";
          str temp_eff = "";

          while (amm.FindChildElem("REFINT"))
```

```
            {
              RefKey = RefKey + "%" + amm.GetChildAttrib("REFID");
              int tempchind = amm.GetChildElemIndex();
              amm.GotoElemIndex(tempchind);
              if (amm.FindChildElem("EFFECT"))
              {
                temp_eff = temp_eff + " " + amm.GetChildAttrib("EFFRG");
              }
              amm.GotoElemIndex(TempInd);
              amm.GotoChildElemIndex(tempchind);
            }
            RefKey = StrMid(RefKey,1,-1);
            if (temp_eff != "")
            {
              eff_b = 0;
              EffRef = temp_eff;
              amm.GotoElemIndex(TempInd);
            }
          }
          else if(amm.FindChildElem("EFFECT"))
          {
            EffRef = amm.GetChildAttrib("EFFRG");
            amm.GotoElemIndex(amm.GetChildElemIndex());
            while (amm.FindChildElem("SBEFF"))
            {
              EffRef = EffRef + " " + amm.GetChildAttrib("EFFRG");
            }
            eff_b = 0;
            amm.GotoElemIndex(TempInd);
          }
        }
      }
      wrList.AddElem("ELEM",rec_dad + "|" + DadKey + "|" + AilRev_
      _+ "|" + RefKey + "|" + EffRef);
      wrList.SetAttrib("REFINT",RefKey);
      wrList.SetAttrib("EFF1",EffRef);
      amm.GotoElemIndex(ActRefInt);
    }
    str savingPath = Path + "\\ReferencesList.xml";
    // Guardamos el resultado en la carpeta en cuestión
    wrList.Save(savingPath);

    return wrList;
}
```

Este código busca cada etiqueta $< REFINT >$ que se encuentre en el manual y, una vez detectada una etiqueta concreta, extrae toda la información acerca de la etiqueta que se ha expuesto antes de presentar el código.

El resultado de este código será un archivo XML que se guardará automáticamente en la carpeta en la que se encuentre la macro únicamente a modo de *backup*. Dicho archivo XML contendrá toda la información expuesta dentro de la etiqueta $< ELEM >$ como texto plano y diferenciada entre sí mediante el separador "|", caracter que se utiliza habitualmente para facilitar la separación en columnas de la información en Excel posteriormente. Además, los datos que serán necesarios para el siguiente paso de extracciones se almacenará en atributos de esta etiqueta para un mejor rendimiento computacional.

Una sección del resultado obtenido sería:

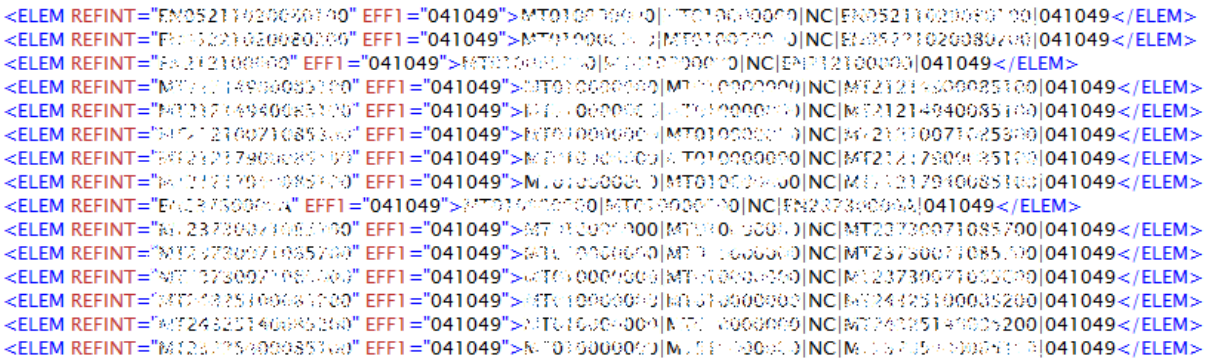


Figura 4.8: Resultado de Paso 1; Detección de hipervínculos en AMM

4.3.3. Paso 2: Ordenación de datos y exclusión de datos corruptos

Hemos ido añadiendo los datos obtenidos al XML de resultado a medida que nos hemos ido encontrando las llamadas a lo largo del AMM. Hemos añadido el atributo *REFINT* por doble motivo: por un lado nos ayudará a ordenar alfabéticamente las llamadas a tareas y, por otro lado, nos va a permitir en el tercer paso hacer sencillas búsquedas sobre el XML de resultado ahorrando tiempo computacional.

Realizar esta ordenación usando el FOXE sería altamente ineficiente a nivel de coste computacional, debido a que no dispone de funciones propias para ordenar la información y pese a programarlas manualmente, jamás se acercaría a los tiempos que se obtienen con Excel.

En este paso necesitamos proporcionar los datos a un archivo Excel para que éste:

1. Ordene las llamadas a tareas por orden alfabético (orden que se corresponde en su mayoría al orden de las tareas en el árbol de tareas del manual).

2. Descarte las llamadas a tareas donde no se proporciona ninguna referencia, por lo que el hipervínculo dará un fallo seguro.

3. Guarde estas llamadas corruptas en el archivo de presentación de datos final

4. Copie los datos no-corruptos a priori para proporcionárselos a la segunda macro de FOXE en el Paso 3.

Crearemos una macro en Excel que ejecute todos estos pasos automáticamente, de modo que al usar el programa, sólo haya que copiar los datos en una columna del Excel y pulsar el botón *Fase*1. Después de ejecutar esta macro, tendremos directamente los datos necesarios copiados en el portapapeles.

El código VBA que ejecuta estas acciones es el siguiente:

```
Sub ordena_primera_extraccion()

Dim lRow As Long
lRow = Cells(Rows.Count, 1).End(xlUp).Row
'Calculamos cuántos registros se han pegado

Dim auxchain As String
auxchain = "A2:A" & lRow
Range(auxchain).Copy
```

```
'Seleccionamos los nuevos registros añadidos

Cells(2, 2).Select
'Seleccionamos celda sobre la que queremos pegar los datos
ActiveCell.PasteSpecial 'Pegamos datos
Columns("B:B").Sort key1:=Range("B2"), order1:=xlAscending, Header:=xlYes
'Ordenamos la columna

Cells(2, 2).Select
Dim inData As String
inData = ActiveCell.Value

Dim i As Integer
i = 1
While InStr(inData, "||") <> 0
'Separamos los casos de referencias que no están correctamente rellenas
  inData = ActiveCell.Offset(i, 0).Value
  i = i + 1
Wend

If i <> 2 Then
'Pegamos los datos y copiamos los que tenemos_
'que llevarnos al Programa 2 de FOXE
  auxchain = "B2:B" & i
  Range(auxchain).Copy
  Worksheets(2).Select
  Cells(2, 1).PasteSpecial
  Worksheets(1).Select

  auxchain = "B" & (i + 1) & ":B" & lRow
  Range(auxchain).Copy
Else
  auxchain = "B2:B" & lRow
  Range(auxchain).Copy
End If

End Sub
```

Pegamos el resultado obtenido en el Paso 1 sobre la primera columna y ejecutamos la primera fase de la macro.

4.3.4. Paso 3: Obtención de efectividades de las tareas llamadas y detección de los errores

Puesto que tenemos ya el XML que debemos introducir a la segunda macro del FOXE, abrimos el software y pegamos los datos un nuevo archivo. A continuación, ejecutamos la macro.

Esta macro debe:

1. A partir de la referencia de la llamada, localizar la tarea en el manual. Si no la encuentra tendremos un fallo de Máster.

2. Si la encuentra, debe obtener la efectividad y el atributo $AIL - REV$.

3. Convertir la efectividad de la llamada D a forma discreta.

4. Convertir la efectividad de la tarea llamada C a forma discreta.

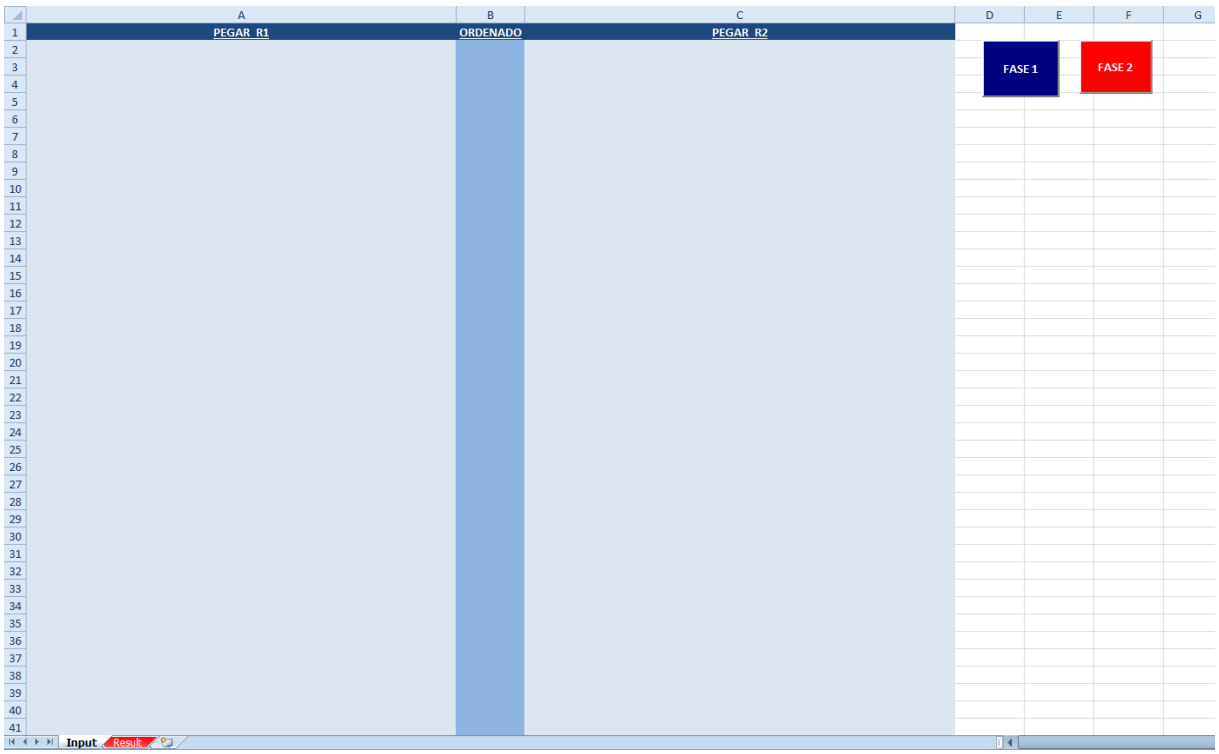


Figura 4.9: Paso 2 antes de ejecutarse

5. Comprobar qué aviones de $D \notin C$.

6. En caso de que haya alguno, apuntarlo en el XML resultado.

7. Devolver los datos.

El XML devuelto debe pegarse en el Excel anterior para ejecutar la Fase 2 y obtener así el resultado final.

El código empleado es el siguiente:

```
CMarkup Second_Func(CMarkup wrList)
{
  str Path = "[insert active folder path]";
  wrList.ResetPos();
  CMarkup amm;
  CMarkup auxamm;
  amm.Load("C:\\Users\\U39750\\Desktop\\AMM_AUG19.sgm");

  while (wrList.FindElem("ELEM"))
  {
    str refint = wrList.GetAttrib("REFINT");
    str ref_eff;
    str ref_ailrev;
    if (StrFind(refint,"%",3) != -1)
    {
      wrList.SetAttrib("REFINT",wrList.GetAttrib("REFINT") + "%");
      while (StrFind(refint,"%",0) != -1)
      {
        str actrefint = StrMid(refint,0,StrFind(refint,"%",0));
        wrList.SetAttrib("REFINT",StrMid(wrList.GetAttrib("REFINT"),_
        _StrFind(refint,"%",0)+1,-1));
        if (actrefint == amm.GetAttrib("KEY"))
        {
```

```
        int ActAmmInd = amm.GetElemIndex();
        ref_ailrev = ref_ailrev + " " + amm.GetAttrib("AIL-REV");

        amm.FindChildElem("EFFECT");
        ref_eff = ref_eff + " " + amm.GetChildAttrib("EFFRG");
        amm.GotoElemIndex(amm.GetChildElemIndex());

        while (amm.FindChildElem("SBEFF"))
        {
          ref_eff = ref_eff + " " + amm.GetChildAttrib("EFFRG");
        }
        wrList.SetData(wrList.GetData()+"|"+ref_ailrev+"|"+ref_eff);
        wrList.SetAttrib("EFF2",ref_eff);
        amm.GotoElemIndex(ActAmmInd);
      }
      else
      {
        amm.ResetPos();
        str str2LF = "//*[@KEY=" + actrefint + "]";
        bool isFound = amm.FindElem(str2LF);

        if (isFound==1)
        {
          int ActAmmInd = amm.GetElemIndex();
          ref_ailrev = ref_ailrev + " " + amm.GetAttrib("AIL-REV");
          amm.FindChildElem("EFFECT");
          ref_eff = ref_eff + " " + amm.GetChildAttrib("EFFRG");
          amm.GotoElemIndex(amm.GetChildElemIndex());

          while (amm.FindChildElem("SBEFF"))
          {
            ref_eff = ref_eff+" "+amm.GetChildAttrib("EFFRG");
          }
          wrList.SetData(wrList.GetData()+"|"+ref_ailrev+-
          -"|"+ref_eff);
          wrList.SetAttrib("EFF2",ref_eff);

          amm.GotoElemIndex(ActAmmInd);
        }
        else
        {
          wrList.SetData(wrList.GetData() + "|" + " ");
          wrList.SetAttrib("EFF2","REFERENCE NOT FOUND");
        }
      }
      refint = wrList.GetAttrib("REFINT");
    }
  }
  else if (refint == amm.GetAttrib("KEY"))
  {
    int ActAmmInd = amm.GetElemIndex();

    ref_ailrev = amm.GetAttrib("AIL-REV");
    amm.FindChildElem("EFFECT");
    ref_eff = amm.GetChildAttrib("EFFRG");
    amm.GotoElemIndex(amm.GetChildElemIndex());

    while (amm.FindChildElem("SBEFF"))
    {
      ref_eff = ref_eff + " " + amm.GetChildAttrib("EFFRG");
```

```
      }
      wrList.SetData(wrList.GetData()+"|"+ref_ailrev+"|"+ref_eff);
      wrList.SetAttrib("EFF2",ref_eff);

      amm.GotoElemIndex(ActAmmInd);
    }
    else
    {
      amm.ResetPos();
      str str2LF = "//*[@KEY=" + refint + "]";
      bool isFound = amm.FindElem(str2LF);

      if (isFound==1)
      {
        int ActAmmInd = amm.GetElemIndex();
        ref_ailrev = amm.GetAttrib("AIL-REV");
        amm.FindChildElem("EFFECT");
        ref_eff = amm.GetChildAttrib("EFFRG");
        amm.GotoElemIndex(amm.GetChildElemIndex());

        while (amm.FindChildElem("SBEFF"))
        {
          ref_eff = ref_eff + " " + amm.GetChildAttrib("EFFRG");
        }
        wrList.SetData(wrList.GetData()+"|"+ref_ailrev+"|"+ref_eff);
        wrList.SetAttrib("EFF2",ref_eff);

        amm.GotoElemIndex(ActAmmInd);
      }
      else
      {
        wrList.SetData(wrList.GetData() + "|" + " ");
        wrList.SetAttrib("EFF2","REFERENCE NOT FOUND");
      }
    }
  }

  CMarkup errorList = Effects_comparer(wrList);
  return errorList;
}
```

El resultado de esta macro será un XML muy similar al anterior. De hecho, simplemente añade datos a dicho XML: añade en el texto plano la efectividad y el atributo *AIL-REV* de la tarea en AMM y añade un nuevo atributo con la efectividad de la tarea en AMM de modo que sea fácilmente accesible para la función *Effects_comparer* que se muestra a continuación:

```
CMarkup Effects_comparer(CMarkup wrList)
{
CMarkup errorList;
wrList.ResetPos();

while (wrList.FindElem("ELEM"))
{
  str eff1 = wrList.GetAttrib("EFF1");
  str eff2 = wrList.GetAttrib("EFF2");

if (StrFind(eff2,"REFERENCE NOT FOUND",0)!=-1)
{
  wrList.SetData(wrList.GetData() + "|" + "REFERENCE NOT FOUND");
```

```
}
else
{
  eff1 = eff_discretizer(eff1);
  // Discretizamos las dos efectividades para poder comparar
  eff2 = eff_discretizer(eff2);

  int i = 0;
  bool hasError = 0;
  while (i<StrLength(eff1))
  // Comparamos si cada valor entre barras de EFF1 está dentro de la EFF2
  {
    str sect = StrMid(eff1,i,StrFind(eff1,"/",1)+1);
    eff1 = StrMid(eff1,StrFind(eff1,"/",1)+1,-1);

    if (StrFind(eff2,sect)==-1)
    // Tenemos que añadir la casuística concreta de FSTA: 16 equivale a 516 + 616 _
    //y así para 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 y 29
    {
      if (sect=="/16/" || sect=="/17/" || sect=="/18/" || sect=="/19/" || _
      _sect=="/20/" || sect=="/21/" || sect=="/22/" || sect=="/23/" || _
      _sect=="/24/" || sect=="/25/" || sect=="/26/" || sect=="/27/" || _
      _sect=="/28/" || sect=="/29/")
      {
        if (StrFind(eff2,StrMid(sect,0,1)+"5"+StrMid(sect,1,-1))==-1 ||_
        _StrFind(eff2,StrMid(sect,0,1)+"6"+StrMid(sect,1,-1))==-1)
        {
          if (hasError==0)
          {
            wrList.SetData(wrList.GetData() + "|" + sect);
            hasError=1;
          }
          else
          {
            wrList.SetData(wrList.GetData() + sect);
          }
        }
      }
      else if (sect=="/516/" || sect=="/517/" || sect=="/518/" _
      _|| sect=="/519/" || sect=="/520/" || sect=="/521/" || _
      _sect=="/522/" || sect=="/523/" || sect=="/524/" || sect=="/525/" _
      _|| sect=="/526/" || sect=="/527/" || sect=="/528/" || sect=="/529/")
      {
        if (StrFind(eff2,StrMid(sect,0,1)+StrMid(sect,2,-1))==-1)
        {
          if (hasError==0)
          {
            wrList.SetData(wrList.GetData() + "|" + sect);
            hasError=1;
          }
          else
          {
            wrList.SetData(wrList.GetData() + sect);
          }
        }
      }
      else if (sect == "/616/" || sect=="/617/" || sect=="/618/" || _
      _sect=="/619/" || sect=="/620/" || sect=="/621/" || sect=="/622/" ||_
      _ sect=="/623/" || sect=="/624/" || sect=="/625/" || sect=="/626/" ||_
      _ sect=="/627/" || sect=="/628/" || sect=="/629/")
```

```
      {
        if (StrFind(eff2,StrMid(sect,0,1)+StrMid(sect,2,-1))==-1)
        {
          if (hasError==0)
          {
            wrList.SetData(wrList.GetData() + "|" + sect);
            hasError=1;
          }
          else
          {
            wrList.SetData(wrList.GetData() + sect);
          }
        }
      }
      else if (hasError==0)
      {
        wrList.SetData(wrList.GetData() + "|" + sect);
        hasError=1;
      }
      else
      {
        wrList.SetData(wrList.GetData() + sect);
      }
     }
    }
     if (hasError == 1)
     {
      errorList.AddElem("ELEM",wrList.GetData());
     }
   }
}

return errorList;
}
```

Esta nueva función contendrá toda la información necesaria para trazar los errores en texto plano separando las unidades de información por "|". En el siguiente paso se podrá ver toda la información que se ha obtenido de manera detallada.

4.3.5. Paso 4: Presentación de datos

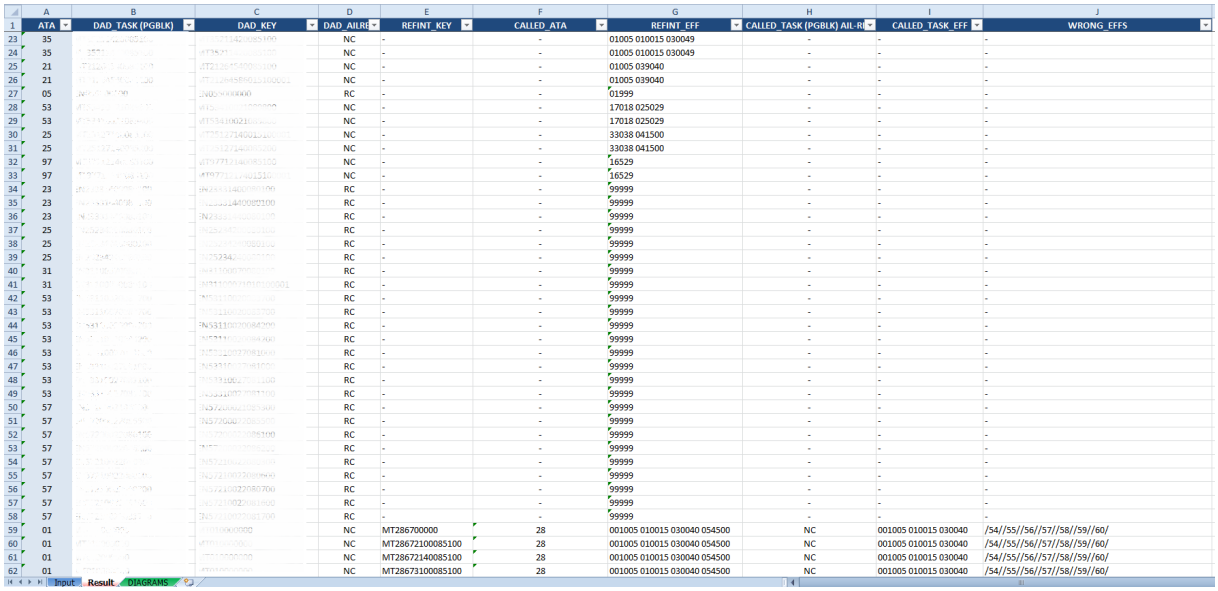
Se ejecuta la Fase 2 del Excel con la siguiente macro:

```
Sub presenta_datos()

Dim lRow As Long
lRow = Cells(Rows.Count, 3).End(xlUp).Row
'Calculamos cuántos registros se han pegado

Dim auxchain          ' Copiamos los nuevos datos introducidos
auxchain = "C2:C" & lRow
Range(auxchain).Copy

Worksheets(2).Select     ' Pegamos los datos en la hoja de resultados
lRow = Cells(Rows.Count, 1).End(xlUp).Row
Cells(lRow + 1, 1).Select
ActiveCell.PasteSpecial

lRow = Cells(Rows.Count, 1).End(xlUp).Row
'Volvemos a contar cuántas celdas hay en la hoja
```

```
' Separamos los casos en columnas distintas
Dim i As Integer
i = 2
While i <= lRow
'Extraemos toda la información de las cadenas de texto pegadas

  Dim b1 As Boolean
  b1 = False

  If (InStr(Cells(i, 1).Value, "||") = 0) Then
    b1 = True
  End If

  Dim edit_str As String
  edit_str = Cells(i, 1).Value
  edit_str = Left(edit_str, Len(edit_str) - 7)
  Dim cutter As Integer
  cutter = InStr(edit_str, ">")
  edit_str = Mid(edit_str, cutter + 1)

  Dim dad_task As String
  cutter = InStr(edit_str, "|")
  dad_task = Mid(edit_str, 1, cutter - 1)
  edit_str = Mid(edit_str, cutter + 1)

  Dim dad_key As String
  cutter = InStr(edit_str, "|")
  dad_key = Mid(edit_str, 1, cutter - 1)
  edit_str = Mid(edit_str, cutter + 1)

  Dim dad_ailrev As String
  cutter = InStr(edit_str, "|")
  dad_ailrev = Mid(edit_str, 1, cutter - 1)
  edit_str = Mid(edit_str, cutter + 1)

  Dim refint_key As String
  Dim refint_eff As String
  Dim called_task_ailrev As String
  Dim called_task_eff As String
  Dim wrong_effs As String
  Dim called_ata
  If b1 = True Then
    cutter = InStr(edit_str, "|")
    refint_key = Mid(edit_str, 1, cutter - 1)
    edit_str = Mid(edit_str, cutter + 1)

    cutter = InStr(edit_str, "|")
    refint_eff = Mid(edit_str, 1, cutter - 1)
    edit_str = Mid(edit_str, cutter + 1)

    cutter = InStr(edit_str, "|")
    called_task_ailrev = Mid(edit_str, 1, cutter - 1)
    edit_str = Mid(edit_str, cutter + 1)

    cutter = InStr(edit_str, "|")
    called_task_eff = Mid(edit_str, 1, cutter - 1)
    edit_str = Mid(edit_str, cutter + 1)

    wrong_effs = edit_str
```

```
    called_ata = Right(Left(refint_key, 4), 2)
  Else
    refint_key = "-"
    called_ata = "-"
    refint_eff = Right(edit_str, Len(edit_str) - 2)
    called_task_ailrev = "-"
    called_task_eff = "-"
    wrong_effs = "-"
  End If

  Range("A:A").NumberFormat = "@"

  'Pegamos la información en columnas separadas
  Cells(i, 1).Value = Right(Left(dad_task, 4), 2)
  Cells(i, 2).Value = dad_task
  Cells(i, 3).Value = dad_key
  Cells(i, 4).Value = dad_ailrev
  Cells(i, 5).Value = refint_key
  Cells(i, 6).Value = called_ata
  Cells(i, 7).Value = refint_eff
  Cells(i, 8).Value = called_task_ailrev
  Cells(i, 9).Value = called_task_eff
  Cells(i, 10).Value = wrong_effs

  Range("A:A").HorizontalAlignment = xlHAlignCenter

  i = i + 1
Wend

Cells(1, 1).Select
End Sub
```

Volviendo al archivo Excel anterior, pegamos los datos en la segunda columna y ejecutamos la Fase 2. Esta macro únicamente separa los datos correctamente y los coloca en las columnas por un lado, y por otro genera los diagramas estadísticos análogos a los presentados anteriormente.



	A	B	C	D	E	F	G	H	I	J
1	ATA	DAD_TASK (PGBLK)	DAD_KEY	DAD_AILRE	REFINT_KEY	CALLED_ATA	REFINT_EFF	CALLED_TASK (PGBLK) AIL-R	CALLED_TASK_EFF	WRONG_EFFS
23	35			NC	-	-	01005 010015 030049	-	-	-
24	35			NC	-	-	01005 010015 030049	-	-	-
25	21			NC	-	-	01005 039040	-	-	-
26	21			NC	-	-	01005 039040	-	-	-
27	05			RC	-	-	01999	-	-	-
28	53			NC	-	-	17018 025029	-	-	-
29	53			NC	-	-	17018 025029	-	-	-
30	25			NC	-	-	33038 041500	-	-	-
31	25			NC	-	-	33038 041500	-	-	-
32	97			NC	-	-	16529	-	-	-
33	97			NC	-	-	16529	-	-	-
34	23			RC	-	-	99999	-	-	-
35	23			RC	-	-	99999	-	-	-
36	23			RC	-	-	99999	-	-	-
37	25			RC	-	-	99999	-	-	-
38	25			RC	-	-	99999	-	-	-
39	25			RC	-	-	99999	-	-	-
40	31			RC	-	-	99999	-	-	-
41	31			RC	-	-	99999	-	-	-
42	53			RC	-	-	99999	-	-	-
43	53			RC	-	-	99999	-	-	-
44	53			RC	-	-	99999	-	-	-
45	53			RC	-	-	99999	-	-	-
46	53			RC	-	-	99999	-	-	-
47	53			RC	-	-	99999	-	-	-
48	53			RC	-	-	99999	-	-	-
49	53			RC	-	-	99999	-	-	-
50	57			RC	-	-	99999	-	-	-
51	57			RC	-	-	99999	-	-	-
52	57			RC	-	-	99999	-	-	-
53	57			RC	-	-	99999	-	-	-
54	57			RC	-	-	99999	-	-	-
55	57			RC	-	-	99999	-	-	-
56	57			RC	-	-	99999	-	-	-
57	57			RC	-	-	99999	-	-	-
58	57			RC	-	-	99999	-	-	-
59	01			NC	MT286700000	28	001005 010015 030040 054500	NC	001005 010015 030040	/54//55//56//57//58//59//60/
60	01			NC	MT28672100085100	28	001005 010015 030040 054500	NC	001005 010015 030040	/54//55//56//57//58//59//60/
61	01			NC	MT28672140085100	28	001005 010015 030040 054500	NC	001005 010015 030040	/54//55//56//57//58//59//60/
62	01			NC	MT28673100085100	28	001005 010015 030040 054500	NC	001005 010015 030040	/54//55//56//57//58//59//60/

Input | Result | DIAGRAMS

Figura 4.10: Paso 4; Pestaña DATA

La pestaña *DIAGRAMS* se formaría del mismo modo que la del apartado anterior y

la macro que generase los diagramas sería prácticamente idéntica, cambiando únicamente las referencias a las ubicaciones de la información en las celdas.

5. Prevención de errores

5.1. ELA

5.1.1. Ejecución de códigos

Para la generación automática del manual será necesario utilizar unas plantillas de base que sirvan para organizar la información, la revisión anterior del manual compuesta por todos los archivos Word que la componen y la macro que la ejecute. La macro deberá ser un procedimiento *Sub* que llame a su vez a una función por documento Word que lo genere. Dicho procedimiento *Sub* debe ser capaz de ir proporcionando las variables de entrada a cada función y, en ocasiones, se usarán datos de salida de una función como entrada de otra.

El programa consistirá entonces en una carpeta que a su vez contiene los siguientes elementos:

- Carpeta donde están contenidos todos los documentos Word de la revisión anterior del manual.
- Carpeta donde están contenidas todas las plantillas.
- Carpeta de resultados donde se almacenarán los nuevos documentos generados.
- Archivo Word que contiene la macro y que tendremos que abrir únicamente para ejecutarla.
- Documento de extension *.xlsx* que contiene la información procedente de Configuración para la elaboración de la nueva revisión del manual. Deberemos pegar este documento en la carpeta una vez lo recibamos para que la macro pueda ejecutarse correctamente.

5.1.2. Cover Page

En primer lugar, tratamos de editar la *Cover Page*.

Este archivo es el más básico de todos. Se trata de la portada del manual en cuestión de cada avión. Está compuesto por los siguientes apartados:

- Logo de la empresa (común para todos los aviones).
- Nombre del manual ELA (común para todos los aviones).
- Nombre de la flota a la que pertenece (por ejemplo, RAAF, RSAF, UAE, ...)[2].
- FSN del avión (exclusivo de cada avión. Es el MSN de la aeronave respecto al programa MRTT).
- MSN del avión (exclusivo de cada avión. Es el MSN de la aeronave respecto a Airbus Civil).

[2]RAAF, RSAF, UAE, RAF2, FSTA, GOST, FAFR, RKAF y MMF son los nombre de los actuales clientes del A330-MRTT. A modo de ejemplo, aunque no es relvante en ningún caso, RAAF son las siglas de *Royal Australian Air Force*

- Copyright que depende del año en que se publique el manual (común a todas las flotas de esa revisión).
- Dirección del departamento *System Support Services*, número Fax a e-mail de contacto (común a todas las aeronaves).
- Revisión del manual, que debe ser una unidad superior al número de revisión de la última publicada para ese manual (exclusivo para cada manual).
- Fecha de revisión en que se publica el manual (depende de la revisión en que se publique).
- Rol Civil o Militar en caso de que el cliente involucrado sea FSTA.

Para poder abordar la edición de este apartado necesitamos:

- Input que nos permita identificar el avión del manual a editar.
- Input que nos permita conocer la fecha de revisión.
- Cover Page anterior para conocer el número de revisión.
- Input sobre el rol de la aeronave si es FSTA.

Con tan sólo conocer el MSN de la aeronave, podemos darle a través del código los demás datos ya que el MSN distingue de manera unívoca a cada avión. Con el MSN podemos conocer el FSN, y la flota. Además, debemos de disponer de la última revisión del manual para poder extraer datos de ella.

Para que este programa pueda ser flexible y admitir pequeños cambios posteriormente, se utiliza una plantilla que podrá editarse de modo que la macro siga podiendo aplicarse. Los datos que varían según los inputs que le demos al programa, pueden encontrarse en la plantilla con nomenclaturas tipo *XXX0NXXX*.

- **XXX01XXX**: Flota.
- **XXX02XXX**: FSN.
- **XXX03XXX**: MSN.
- **XXX04XXX**: Fecha del copyright.
- **XXX05XXX**: Número de revisión.
- **XXX06XXX**: Fecha de revisión.
- **XXX07XXX**: Rol, en caso de ser FSTA.

De este modo, la plantilla se encontrará en la ruta *\\ELA_Generator\Plantillas\01_Cover Page.doc*. Su aspecto será el siguiente:

El código que nos permite convertir la plantilla en el documento final es el siguiente:

Electric Load Analysis

ELA

XXX01XXX FSN XXX02XXX

(AIRBUS MSN XXX03XXX)

XXX07XXX

AIRBUS DEFENCE AND SPACE, S.A.U.
Avenida de Aragón 404, 28022 Madrid, SPAIN
System Support Services
Technical Information and Data
Fax (34) 954-59-3623
E-mail = AD.TID.Support@airbus.com

Revision Number: XXX06XXX
XXX05XXX

Figura 5.1: Plantilla de COVER

```
Function Creador_cover(rev As String, MSN As String,
FSN As String, fleet As String, CoM As String, wPath As String)

Dim nrv_str As String
'Variable de salida de la función (el número de la siguiente
'revisión del documento como cadena de texto)

'Buscamos el número de revisión anterior del ELA
''Abrimos el documento anterior de MFM_Anterior
Dim oldPath As String
oldPath = wPath + "\MFM anterior\01_Cover Page.doc"
Documents.Open (oldPath)
Documents("01_Cover Page.doc").Select

''Buscamos en las cajas de texto aquella que ponga
''"Revision Number"
Dim txtb As Shape
For Each txtb In ActiveDocument.Shapes
  If txtb.Type = msoTextBox Then
    txtb.Select
    Selection.ShapeRange.TextFrame.TextRange.Select

    If Left(Selection.Text, 16) = "Revision Number:" Then
      Dim orv As Integer
      If Left(Right(Selection.Range, 4), 2) = "on" Then
        nrv_str = "01"
      Else
        orv = CInt(Left(Right(Selection.Range, 4), 2))
        Dim nrv As Integer
        nrv = orv + 1

        nrv_str = CStr(nrv)
        If Len(nrv_str) = 1 Then
          nrv_str = "0" + nrv_str
        End If
      End If
    End If
  End If
Next
ActiveDocument.Close _
SaveChanges:=wdDoNotSaveChanges

'Sustituimos datos sobre la plantilla
''Abrimos plantilla
Dim newPath As String
newPath = wPath + "\Plantillas\01_Cover Page.doc"
Documents.Open (newPath)
Documents("01_Cover Page.doc").Select

''Buscamos y sustituimos en los 7 campos
Dim rev_yr As String
rev_yr = "20" + Right(rev, 2)
```

```
For Each txtb In ActiveDocument.Shapes
  If txtb.Type = msoTextBox Then

    txtb.Select
    Selection.ShapeRange.TextFrame.TextRange.Select
    With Selection.Find
      .Forward = True
      .Wrap = wdFindStop
      .Text = "XXX01XXX"
      .Execute FindText:="XXX01XXX", ReplaceWith:=fleet
    End With

    txtb.Select
    Selection.ShapeRange.TextFrame.TextRange.Select
    With Selection.Find
      .Forward = True
      .Wrap = wdFindStop
      .Text = "XXX02XXX"
      .Execute FindText:="XXX02XXX", ReplaceWith:=FSN
    End With

    txtb.Select
    Selection.ShapeRange.TextFrame.TextRange.Select
    With Selection.Find
      .Forward = True
      .Wrap = wdFindStop
      .Text = "XXX03XXX"
      .Execute FindText:="XXX03XXX", ReplaceWith:=MSN
    End With

    txtb.Select
    Selection.ShapeRange.TextFrame.TextRange.Select
    With Selection.Find
      .Forward = True
      .Wrap = wdFindStop
      .Text = "XXX04XXX"
      .Execute FindText:="XXX04XXX", ReplaceWith:=rev_yr
    End With

    txtb.Select
    Selection.ShapeRange.TextFrame.TextRange.Select
    With Selection.Find
      .Forward = True
      .Wrap = wdFindStop
      .Text = "XXX05XXX"
      .Execute FindText:="XXX05XXX", ReplaceWith:=nrv_str
    End With

    txtb.Select
    Selection.ShapeRange.TextFrame.TextRange.Select
    Selection.Find.Execute FindText:="XXX06XXX"
    If Selection.Find.Found = True Then
      Selection.Range = "Revision Date: " + rev
```

```
    End If

    txtb.Select
    Selection.ShapeRange.TextFrame.TextRange.Select
    With Selection.Find
      .Forward = True
      .Wrap = wdFindStop
      .Text = "XXX07XXX"
      .Execute FindText:="XXX07XXX", ReplaceWith:=CoM
    End With
  End If
Next

''Guardamos
Dim finalPath As String
finalPath = wPath +
"\Resultados\01-Manual Front Matter\01_Cover Page.doc"
ActiveDocument.SaveAs2 FileName:=finalPath, _
FileFormat:=wdFormatDOC
Creador_cover = nrv_str
'La salida de esta función nos da la revisión
'siguiente para el apartado de TOC

''Cerramos .doc
Documents("01_Cover Page.doc").Close

End Function
```

5.1.3. Record of Revisions

El *Record of Revisions* es un documento que recoge la fecha de todas las revisiones que ha habido del manual en cuestión y el motivo por el que se lanzó a revisión dicho manual.

En este caso nos nutriremos también de una plantilla que incluya el encabezado tipo, la tabla de información que se encuentra en el cuerpo del documento y el pie de página. Tenemos:

- Logo de la empresa (encabezado) (común a todos los aviones).
- Efectividad tipo FSN y tipo MSN (pie de página) (exclusivo de cada avión).
- Fecha de revisión en la que se emite el manual (pie de página) (depende de la revisión en la que se esté publicando)
- Tabla con el contenido principal del documento (cuerpo del documento) (depende de cada avión). Dividida en tres columnas: *Revision Number*, que debe enumerarse hasta el número de revisión que aparece en la nueva *Cover Page*; *Issue Date*, con la fecha de la revisión de cada issue del manual; y *Reason for Change*, con el motivo por el que se lanzó a producción el manual en cada revisión.

Necesitaremos entonces como entradas nuevas para este documento:

- *Record of Revision* de la revisión anterior.

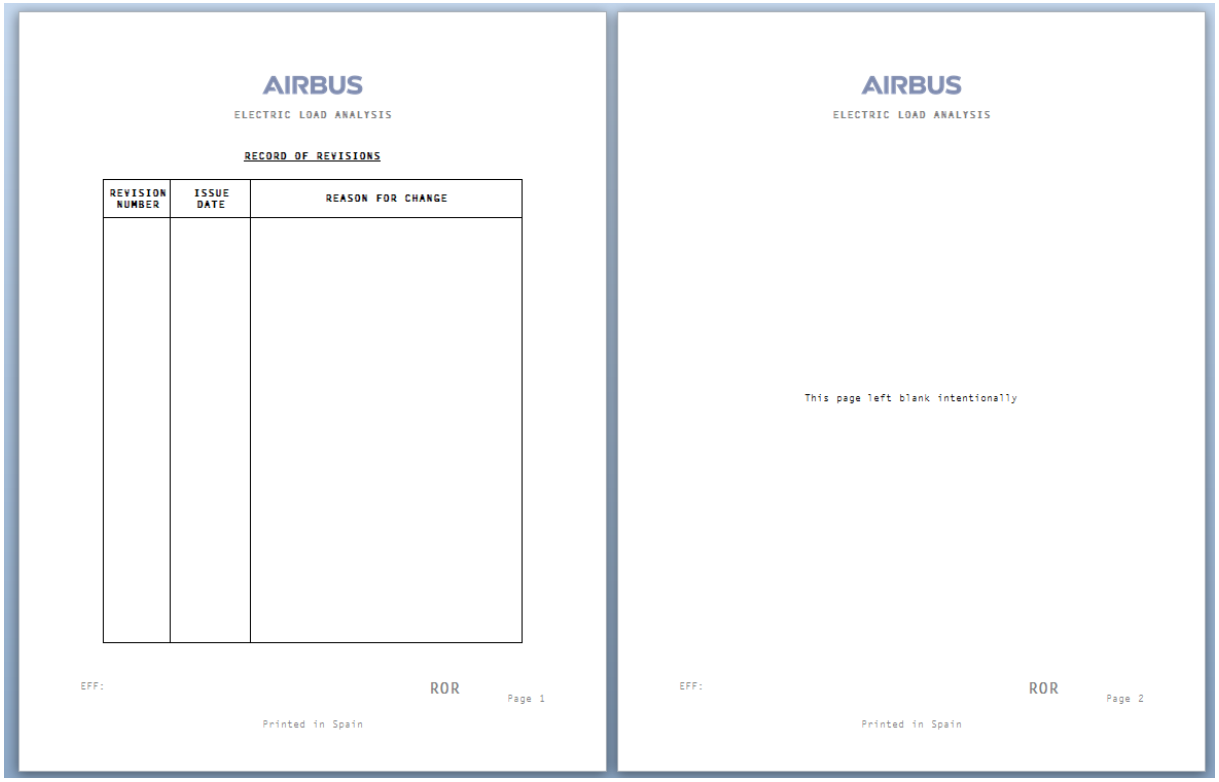


Figura 5.2: Plantilla de Record of Revisions

- Razón para la actual edición y publicación del manual.

La tabla que presenta la plantilla contiene 19 filas, es decir, es capaz de albergar hasta 19 revisiones. Este manual es el que menos revisiones debe tener debido a que no se actualiza periódicamente, como se comentaba anteriormente, sino que se actualiza la información cuando el equipo se percata de que hay algún tipo de error. Es decir, como no se van incluyendo criterios trimestralmente, es altamente improbable que un ELA llegue a necesitar 20 revisiones alguna vez. Por ello, vamos a plantear que no se llegue a necesitar nunca tantas revisiones (hipótesis muy factible). Como el número de páginas debe ser par, se incluye una segunda página con el texto *"This page left blank intentionally"*.

El programa debe ser capaz de leer los datos del antiguo *Record of Revision*, pegarlo en el nuevo archivo y añadir la nueva línea con los datos correspondientes. La plantilla se encontrará en la ruta *\\ELA_Generator\Plantillas\02_Record of revisions_p.doc*. El motivo de ese *"_p"* al final se debe a que a lo largo de la ejecución de la macro que ahora vamos a presentar será necesario tener también abierto el *Record of Revision* de la revisión anterior y no queremos tener abiertos dos archivos del mismo tipo y con el mismo nombre.

El código es el siguiente:

```
Function Creador_ROR(rev As String, MSN As String, FSN As String,
fleet As String, CoM As String, chg_reason As String,
nrv_str As String, wPath As String)

'Abrimos los doscumentos Word
Dim prevPath As String
prevPath = wPath + "\MFM anterior\02_Record of Revisions.doc"
Documents.Open (prevPath)
Dim planPath As String
```

```
planPath = wPath + "\Plantillas\02_Record of Revisions_p.doc"
Documents.Open (planPath)

'Pasamos información de un documento a otro
Dim tabRun As Boolean
tabRun = True
Dim iRun As Integer
iRun = 2
While tabRun = True
  If Documents("02_Record of Revisions.doc").Tables(1).Cell(iRun, 1)._
  Range <> Chr(13) & Chr(7) Then

    Documents("02_Record of Revisions_p.doc").Tables(1)._
    Cell(iRun, 1).Range = Documents("02_Record of Revisions.doc")._
    Tables(1).Cell(iRun, 1).Range

    Documents("02_Record of Revisions_p.doc").Tables(1)._
    Cell(iRun, 2).Range = Documents("02_Record of Revisions.doc")._
    Tables(1).Cell(iRun, 2).Range

    Documents("02_Record of Revisions_p.doc").Tables(1)._
    Cell(iRun, 3).Range = Documents("02_Record of Revisions.doc")._
    Tables(1).Cell(iRun, 3).Range

    iRun = iRun + 1
  Else
    tabRun = False
  End If
Wend

Documents("02_Record of Revisions_p.doc").Tables(1).Cell(iRun, 1)._
Range = nrv_str

Documents("02_Record of Revisions_p.doc").Tables(1).Cell(iRun, 2)._
Range = rev
Documents("02_Record of Revisions_p.doc").Tables(1).Cell(iRun, 3)._
Range = chg_reason

'Rellenamos pie de página
Dim str_mfsn As String
If CoM = "" Then
  str_mfsn = "FSN " & FSN & " (MSN " & MSN & ")"
Else
  If CoM = "MILITARY" Then
    str_mfsn = "FSN " & FSN & " (MSN " & MSN & ") VYG-KC"
  Else
    str_mfsn = "FSN " & FSN & " (MSN " & MSN & ") VYG-CI"
  End If
End If

Documents("02_Record of Revisions_p.doc").Sections(1)._
Footers(wdHeaderFooterPrimary).Range.Tables(1).Range._
Cells(2).Range.InsertAfter str_mfsn
```

```
Documents("02_Record of Revisions_p.doc").Sections(1).
Footers(wdHeaderFooterPrimary).Range.Tables(1).Range.
Cells(6).Range.InsertAfter rev

'Cerramos .doc
Documents("02_Record of Revisions.doc").Close

'Guardamos
Dim finalPath As String
finalPath = wPath + "\Resultados\01-Manual Front Matter\
02_Record of Revisions.doc"

Documents("02_Record of Revisions_p.doc").SaveAs2 _
FileName:=finalPath, FileFormat:=wdFormatDOC

'Cerramos .doc
Documents("02_Record of Revisions.doc").Close

End Function
```

5.1.4. Tables

Este archivo es el más complejo de los que componen el manual y el que contiene toda la información técnica del manual: presenta, para cada bus de la aeronave, la potencia máxima que es capaz de proporcionar y, por tanto, qué equipos pueden conectarse al mismo tiempo a dicho bus dependiendo de en qué fase de vuelo nos encontremos o qué rol este adoptando la aeronave.

Este documento consta en su cabecera del logo de la empresa y en el pie de página los mismo datos que contiene, por ejemplo, el *Record of Revision*.

En el cuerpo del documento vamos a encontrar tantas tablas como hojas tenga el documento Excel que contenga la información. A su vez, cada tabla va a tener una introducción que presente el texto *" ELECTRICAL LOAD SUB-BUSBAR "*, seguido del nombre del bus (que será generalmente un número FIN) y seguido de la configuración de la aeronave a la que quede referida la información: si lleva PODS, BOOM, CONSOLE, etc.

Las tablas estarán formadas por una cabecera que explica el contenido de cada columna, el contenido de cada equipo y el resumen o suma total de cada columna.

Las columnas tendrán formatos distintos. Asimismo, será necesario que el contenido resumen de cada tabla quede perfectamente integrado al final de la última página que abarque el documento y de modo completo. En caso de que esto no pudiera llevarse a cabo, se procedería a realizar un salto de página para que se reflejase este contenido en la siguiente página.

La manera de iterar consistirá en ir abriendo cada una de las hojas del archivo Excel, copiar los datos, crear la siguiente tabla en el archivo Word, pegar la información, formatearla y situar finalmente el contenido resumen en el correcto lugar.

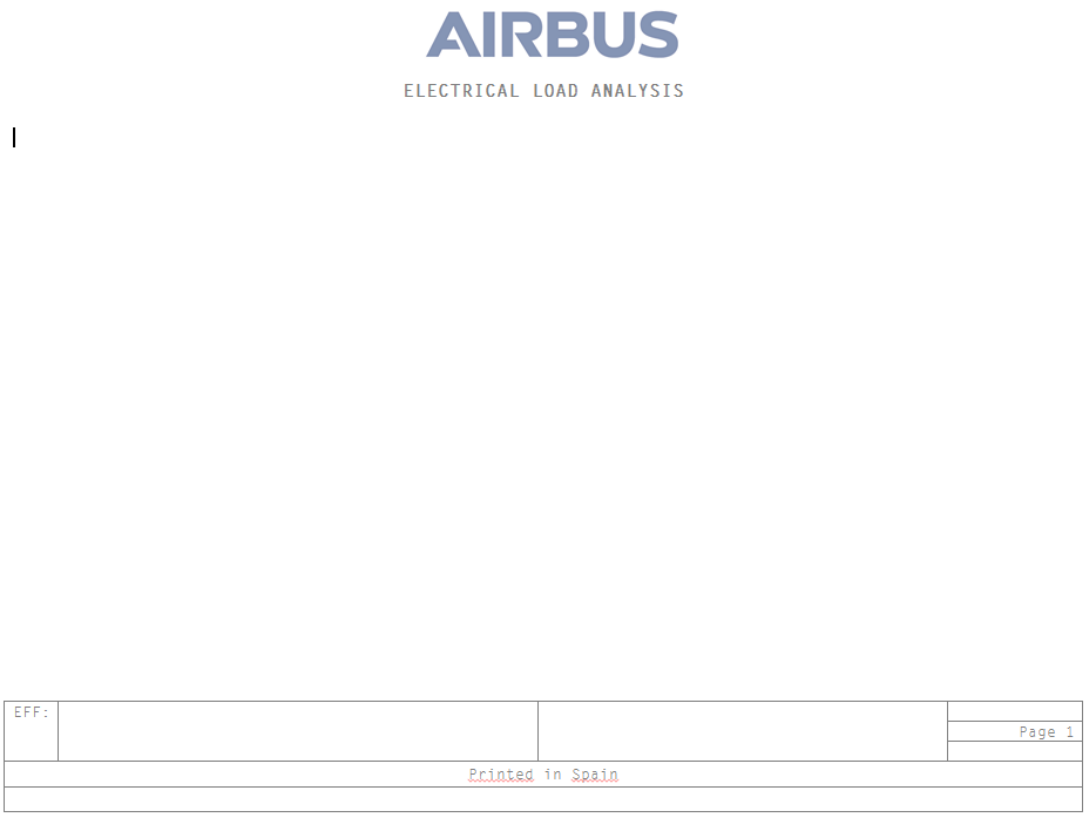


Figura 5.3: Plantilla de Tables

Será necesario hacer uso del paquete de programación VBA para Word que venimos usando para el manual ELA y, adicionalmente, el paquete de programación VBA para Excel con objeto de poder manejar correctamente la información de este archivo cuando lo abramos.

Nos quedaría el siguiente código:

```
Function Creador_tablas(rev As String, MSN As String, FSN As String, _
_fleet As String, CoM As String)

'Obtenemos dirección del archivo .docm
Dim wPath As String 'Variable a pasar: el path del archivo base
wPath = ActiveDocument.Path

'Ponemos el encabezado
Documents.Open (wPath & "\Plantillas\00_Tables.doc")
Documents("00_Tables.doc").Select
ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range._
_Tables(1).Range.Copy
Documents("Executer.docm").Select
ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range.Select
Selection.Paste


'Ponemos el pie de página
Documents("00_Tables.doc").Select
ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary)._
_Range.Tables(1).Range.Copy
Documents("Executer.docm").Select
ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary)._
_Range.Paste
ActiveDocument.Paragraphs.Last.Range.Delete
Documents("00_Tables.doc").Close

Documents("Executer.docm").Select
Dim str_mfsn As String
If CoM = "" Then
  str_mfsn = "FSN " & FSN & " (MSN " & MSN & ")"
Else
  If CoM = "MILITARY" Then
    str_mfsn = "FSN " & FSN & " (MSN " & MSN & ") VYG-KC"
  Else
    str_mfsn = "FSN " & FSN & " (MSN " & MSN & ") VYG-CI"
  End If
End If
ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary).Range._
_Tables(1).Range.Cells(2).Range.InsertAfter str_mfsn
ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary).Range._
_Tables(1).Range.Cells(6).Range.InsertAfter rev
ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary).Range._
_Tables(1).Borders.Enable = False

'Ponemos orientación apaisada del .docm
ActiveDocument.PageSetup.Orientation = wdOrientLandscape
```

```
'Contamos el número de páginas del .docm
Dim nPages As Integer
nPages = ActiveDocument.ComputeStatistics(wdStatisticPages)

'Abrimos .xlsx asociado
Dim oExcel As Excel.Application
Dim oWB As Workbook
Set oExcel = New Excel.Application
Dim exName As String
exName = wPath & "\TablaData.xlsx"
'Marcamos la ruta del .xlsx a partir de la ruta del .docm
Set oWB = oExcel.Workbooks.Open(exName)
oExcel.Visible = True

'Contamos el número de hojas del .xlsx
Dim exNsheets As Integer
exNsheets = Excel.Workbooks("TablaData.xlsx").Sheets.Count

'Generamos índice contadores de tablas(Word) o sheets(Excel)
Dim iwo As Integer
iwo = 1
Dim iexc As Integer
If Excel.Workbooks("TablaData.xlsx").Sheets(1).Name = _
_"AC SIDE MODIFIED" Then
  iexc = 2
Else
  iexc = 1
End If

While iexc <= exNsheets
'Creamos tabla cabecera del .docm
  Excel.Workbooks("TablaData.xlsx").Worksheets(iexc).Select
  Dim sheetName As String
  sheetName = Excel.ActiveSheet.Name

  Documents("Executer.docm").Select

  Set myRange = ActiveDocument.Paragraphs.Last.Range
  ActiveDocument.Tables.Add Range:=myRange, NumRows:=1, NumColumns:=3
  ActiveDocument.Tables(iwo).Borders.Enable = False

  ActiveDocument.Paragraphs.Last.Range.Select
  ActiveDocument.Paragraphs.Add

  'Rellenamos tabla cabecera del .docm
  ''Primera columna
  Documents("Executer.docm").Tables(iwo).Cell(1, 1).Range._
  _InsertAfter "ELECTRICAL LOAD SUB-BUSBAR"
  ActiveDocument.Tables(iwo).Columns(1).Width = InchesToPoints(5.6)
  ActiveDocument.Tables(iwo).Rows(1).Height = InchesToPoints(0.319)
  ActiveDocument.Tables(iwo).Cell(1, 1).VerticalAlignment = _
  _wdCellAlignVerticalCenter
```

```
 ActiveDocument.Tables(iwo).Cell(1, 1).Range.ParagraphFormat._
 _Alignment =wdAlignParagraphRight

''Segunda columna
 Dim FIN As String
 FIN = Mid(sheetName, 1, InStr(1, sheetName, " ") - 1)
 Documents("Executer.docm").Tables(iwo).Cell(1, 2).Range._
 _InsertAfter FIN
 ActiveDocument.Tables(iwo).Columns(2).Width = InchesToPoints(1.736)
 ActiveDocument.Tables(iwo).Cell(1, 2).VerticalAlignment = _
 _wdCellAlignVerticalCenter
 ActiveDocument.Tables(iwo).Cell(1, 2).Range.ParagraphFormat._
 _Alignment = wdAlignParagraphCenter

''Tercera columna
 Dim Kind As String
 While InStr(1, sheetName, ")") <> 0 Or InStr(1, sheetName, "(") <> 0
   If InStr(1, sheetName, ")") <> 0 Then
     sheetName = Mid(sheetName, 1, InStr(1, sheetName, ")") - 1) &_
     _ Mid(sheetName, InStr(1, sheetName, ")") + 1)
   Else
     sheetName = Mid(sheetName, 1, InStr(1, sheetName, "(") - 1) &_
     _ Mid(sheetName, InStr(1, sheetName, "(") + 1)
   End If
 Wend
 Kind = Mid(sheetName, InStr(1, sheetName, " ") + 1)
 If Kind = "MAX" Then
   Documents("Executer.docm").Tables(iwo).Cell(1, 3).Range._
   _InsertAfter "MAXI"
 ElseIf Kind = "OPE" Then
   Documents("Executer.docm").Tables(iwo).Cell(1, 3).Range._
   _InsertAfter "OPERATIONAL"
 Else
   Documents("Executer.docm").Tables(iwo).Cell(1, 3).Range._
   _InsertAfter Kind
 End If
 ActiveDocument.Tables(iwo).Columns(3).Width = InchesToPoints(1.22)
 ActiveDocument.Tables(iwo).Cell(1, 3).VerticalAlignment =_
 _ wdCellAlignVerticalCenter
 ActiveDocument.Tables(iwo).Cell(1, 3).Range.ParagraphFormat._
 _Alignment = wdAlignParagraphLeft

 ''Total tabla
 ActiveDocument.Tables(iwo).Rows(1).Range.Font.Name = "Letter Gothic"
 ActiveDocument.Tables(iwo).Rows(1).Range.Font.Size = 11

 ActiveDocument.Tables(iwo).Columns(1).Select
 Selection.Borders(wdBorderBottom).LineStyle = wdLineStyleSingle
 Selection.Borders(wdBorderTop).LineStyle = wdLineStyleSingle
 ActiveDocument.Tables(iwo).Columns(2).Select
 Selection.Borders(wdBorderBottom).LineStyle = wdLineStyleSingle
 Selection.Borders(wdBorderTop).LineStyle = wdLineStyleSingle
```

```
ActiveDocument.Tables(iwo).Rows.Alignment = wdAlignRowCenter

'Creamos matriz con datos del .xlsx
Dim lRow As Integer
Excel.Workbooks("TablaData.xlsx").Worksheets(iexc).Activate
lRow = Excel.Workbooks("TablaData.xlsx").Worksheets(iexc).
_Cells(Rows.Count, 1).End(xlUp).Row

ReDim MatData(lRow, 17) As String
Dim i As Integer

Dim j As Integer
i = 1

While i <= lRow
  j = 1
  While j <= 17
    MatData(i, j) = Excel.Worksheets(iexc).Cells(i, j).Value
    If IsNumeric(MatData(i, j)) And j > 6 Then
      MatData(i, j) = Round(MatData(i, j), 1)
      If Int(MatData(i, j)) - MatData(i, j) = 0 Then
        MatData(i, j) = MatData(i, j) & ",0"
      End If
    End If
    j = j + 1
  Wend
  i = i + 1
Wend

'Creamos tabla en el archivo .docm
Documents("Executer.docm").Select
Set myRange = ActiveDocument.Paragraphs.Last.Range
ActiveDocument.Tables.Add Range:=myRange, NumRows:=lRow,
_NumColumns:=17
ActiveDocument.Tables(iwo + 1).Borders.Enable = False

ActiveDocument.Paragraphs.Last.Range.Select
ActiveDocument.Paragraphs.Add

'Rellenamos contenido de tabla 2
i = 1

While i <= lRow
  j = 1
  While j <= 17
    Documents("Executer.docm").Tables(iwo + 1).Cell(i, j).
    _Range.InsertAfter MatData(i, j)
    j = j + 1
  Wend
  i = i + 1
Wend

'Ajustamos el tamaño de las columnas
```

```
ActiveDocument.Tables(iwo + 1).Columns(1).Width = _
_InchesToPoints(0.634)
ActiveDocument.Tables(iwo + 1).Columns(2).Width = _
_InchesToPoints(0.2)
ActiveDocument.Tables(iwo + 1).Columns(3).Width = _
_InchesToPoints(0.2)
ActiveDocument.Tables(iwo + 1).Columns(4).Width = _
_InchesToPoints(0.51)
ActiveDocument.Tables(iwo + 1).Columns(5).Width = _
_InchesToPoints(0.37)
ActiveDocument.Tables(iwo + 1).Columns(6).Width = _
_InchesToPoints(2.5)
ActiveDocument.Tables(iwo + 1).Columns(7).Width = _
_InchesToPoints(0.7)
ActiveDocument.Tables(iwo + 1).Columns(8).Delete
ActiveDocument.Tables(iwo + 1).Columns(8).Width = _
_InchesToPoints(0.62)
ActiveDocument.Tables(iwo + 1).Columns(9).Width = _
_InchesToPoints(0.62)
ActiveDocument.Tables(iwo + 1).Columns(10).Width = _
_InchesToPoints(0.62)
ActiveDocument.Tables(iwo + 1).Columns(11).Width = _
_InchesToPoints(0.62)
ActiveDocument.Tables(iwo + 1).Columns(12).Width = _
_InchesToPoints(0.62)
ActiveDocument.Tables(iwo + 1).Columns(13).Width = _
_InchesToPoints(0.62)
ActiveDocument.Tables(iwo + 1).Columns(14).Width = _
_InchesToPoints(0.62)
ActiveDocument.Tables(iwo + 1).Columns(15).Width = _
_InchesToPoints(0.62)
ActiveDocument.Tables(iwo + 1).Columns(16).Width = _
_InchesToPoints(0.62)

'Ajustamos la fuente de las tablas
ActiveDocument.Tables(iwo + 1).Rows(1).Range.Font.Size = 9
ActiveDocument.Tables(iwo + 1).Rows(1).Range.Font.Bold = False
ActiveDocument.Tables(iwo + 1).Rows(1).Range.Font.Italic = False
ActiveDocument.Tables(iwo + 1).Rows(1).Range.Font.Underline = False
ActiveDocument.Tables(iwo + 1).Rows(1).Range.Font._
_Name = "Letter Gothic"

ActiveDocument.Tables(iwo + 1).Cell(1, 1).VerticalAlignment _
_= wdCellAlignVerticalCenter
ActiveDocument.Tables(iwo + 1).Cell(1, 1).Range.ParagraphFormat._
_Alignment= wdAlignParagraphCenter

ActiveDocument.Tables(iwo + 1).Cell(1, 2).VerticalAlignment _
_= wdCellAlignVerticalTop
ActiveDocument.Tables(iwo + 1).Cell(1, 2).Range.ParagraphFormat._
_Alignment = wdAlignParagraphLeft

ActiveDocument.Tables(iwo + 1).Cell(1, 3).VerticalAlignment _
```

```
_= wdCellAlignVerticalBottom
ActiveDocument.Tables(iwo + 1).Cell(1, 3).Range.ParagraphFormat._
_Alignment = wdAlignParagraphRight

ActiveDocument.Tables(iwo + 1).Cell(1, 4).VerticalAlignment _
_= wdCellAlignVerticalCenter
ActiveDocument.Tables(iwo + 1).Cell(1, 4).Range.ParagraphFormat._
_Alignment = wdAlignParagraphCenter

ActiveDocument.Tables(iwo + 1).Cell(1, 5).VerticalAlignment _
_= wdCellAlignVerticalCenter
ActiveDocument.Tables(iwo + 1).Cell(1, 5).Range.ParagraphFormat._
_Alignment = wdAlignParagraphCenter

ActiveDocument.Tables(iwo + 1).Cell(1, 6).VerticalAlignment _
_= wdCellAlignVerticalCenter
ActiveDocument.Tables(iwo + 1).Cell(1, 6).Range.ParagraphFormat._
_Alignment = wdAlignParagraphLeft

Dim k As Integer
k = 7

While k <= 16
  ActiveDocument.Tables(iwo + 1).Cell(1, k).VerticalAlignment _
  _= wdCellAlignVerticalCenter
  ActiveDocument.Tables(iwo + 1).Cell(1, k).Range.ParagraphFormat._
  _Alignment= wdAlignParagraphCenter

  k = k + 1
Wend

k = 2
While k <= lRow
  ActiveDocument.Tables(iwo + 1).Rows(k).Range.Font.Size = 8
  ActiveDocument.Tables(iwo + 1).Rows(k).Range.Font.Name = _
  _"Letter Gothic"
  ActiveDocument.Tables(iwo + 1).Rows(k).Height = _
  _InchesToPoints(0.1653543)
  ActiveDocument.Tables(iwo + 1).Rows(k).Range.Font.Bold = False
  ActiveDocument.Tables(iwo + 1).Rows(k).Range.Font.Italic = False
  ActiveDocument.Tables(iwo + 1).Rows(k).Range.Font._
  _Underline = False

  Dim q As Integer
  q = 1
  While q < 6
    ActiveDocument.Tables(iwo + 1).Cell(k, q).VerticalAlignment _
    _= wdCellAlignVerticalCenter
    ActiveDocument.Tables(iwo + 1).Cell(k, q).Range._
    _ParagraphFormat.Alignment = wdAlignParagraphCenter
    q = q + 1
  Wend
```

```
    ActiveDocument.Tables(iwo + 1).Cell(k, q).VerticalAlignment _
    _= wdCellAlignVerticalCenter
    ActiveDocument.Tables(iwo + 1).Cell(k, q).Range._
    _ParagraphFormat.Alignment = wdAlignParagraphLeft
    q = q + 1

    While q < 17
       ActiveDocument.Tables(iwo + 1).Cell(k, q).VerticalAlignment _
       _= wdCellAlignVerticalCenter
       ActiveDocument.Tables(iwo + 1).Cell(k, q).Range._
       _ParagraphFormat.Alignment = wdAlignParagraphRight
      q = q + 1
    Wend

    k = k + 1
Wend

ActiveDocument.Tables(iwo + 1).Rows.Alignment _
_= wdAlignRowCenter

ActiveDocument.Tables(iwo + 1).Rows(1).Select

Selection.Borders(wdBorderBottom).LineStyle = _
_wdLineStyleDashLargeGap
Selection.Borders(wdBorderTop).LineStyle = _
_wdLineStyleDashLargeGap

Dim lblank As Boolean
lblank = False
Dim qv As Integer
qv = lRow

While lblank = False
    If ActiveDocument.Tables(iwo + 1).Cell(qv, 16).Range = Chr(13)_
    _ & Chr(7) Or qv < 2 Then
      lblank = True
      qv = qv + 1
    Else
      qv = qv - 1
    End If
Wend

Dim sp_row As Integer
sp_row = qv - 1

While qv <= lRow
    ActiveDocument.Tables(iwo + 1).Rows(qv).Select
    Selection.Borders(wdBorderBottom).LineStyle = _
    _wdLineStyleDashLargeGap
    Selection.Borders(wdBorderTop).LineStyle = _
    _wdLineStyleDashLargeGap

    ActiveDocument.Tables(iwo + 1).Cell(qv, 1).Range.Copy
```

```
    ActiveDocument.Tables(iwo + 1).Cell(qv, 6).Range.Paste
    ActiveDocument.Tables(iwo + 1).Cell(qv, 6).Range.ParagraphFormat._
    _Alignment = wdAlignParagraphLeft
    ActiveDocument.Tables(iwo + 1).Cell(qv, 2).Range.Copy
    ActiveDocument.Tables(iwo + 1).Cell(qv, 1).Range.Paste

    qv = qv + 1
  Wend

  'Comprobamos si el tamaño vertical de la tabla está ajustado_
  '_respecto al .docm
  Dim nnp As Integer
  nnp = ActiveDocument.ComputeStatistics(wdStatisticPages)
  Dim isOk As Boolean
  isOk = False

  ''Comprobación de si está bien de entrada
  If nnp > nPages Then
    ActiveDocument.Tables(iwo + 1).Rows(ActiveDocument.Tables_
    _(iwo + 1).Rows.Count).Range.Copy
    ActiveDocument.Tables(iwo + 1).Rows(ActiveDocument.Tables_
    _(iwo + 1)._
    _Rows.Count).Delete

    If ActiveDocument.ComputeStatistics(wdStatisticPages)_
    _ = nnp - 1 Then
      isOk = True
    End If

    ActiveDocument.Tables(iwo + 1).Rows.Add
    ActiveDocument.Tables(iwo + 1).Rows(ActiveDocument.Tables_
    _(iwo + 1).Rows.Count).Range.Paste
    ActiveDocument.Tables(iwo + 1).Rows(ActiveDocument.Tables_
    _(iwo + 1).Rows.Count).Delete
  End If
  ''Si no está bien tenemos que empezar a añadir líneas
  While isOk = False
    ActiveDocument.Tables(iwo + 1).Rows.Add (ActiveDocument._
    _Tables(iwo + 1).Rows(sp_row))
    ActiveDocument.Tables(iwo + 1).Rows(sp_row + 1)._
    _Borders(wdBorderTop).LineStyle = wdLineStyleNone
    Dim Snnp As Integer
    Snnp = ActiveDocument.ComputeStatistics(wdStatisticPages)

    If Snnp = nnp + 1 Then
      isOk = True
    End If
  Wend

  iexc = iexc + 1
  iwo = iwo + 2
Wend
```

```
'Eliminamos última página
ActiveDocument.Bookmarks("\EndOfDoc").Range.Select
ActiveDocument.Bookmarks("\Page").Range.Delete

'Guardamos los archivos tabla generados
Dim str2save As String
dateXfile = "20" + Right(rev, 2)
If Left(rev, 3) = "Feb" Then
  dateXfile = dateXfile + "02"
ElseIf Left(rev, 3) = "May" Then
  dateXfile = dateXfile + "05"
ElseIf Left(rev, 3) = "Aug" Then
  dateXfile = dateXfile + "08"
ElseIf Left(rev, 3) = "Nov" Then
  dateXfile = dateXfile + "11"
End If

If CoM = "" Then
  str2save = wPath + "\Resultados\ELA_" + fleet + "_FSN" + FSN +_
  _ "_MSN" + "_" + dateXfile + ".rtf"
Else
  If CoM = "MILITARY" Then
    str2save = wPath + "\Resultados\ELA_" + fleet + "_FSN" + FSN +_
    _ "_MSN" + "_VYG-KC_" + dateXfile + ".rtf"
  Else
    str2save = wPath + "\Resultados\ELA_" + fleet + "_FSN" + FSN +_
    º_ "_MSN" + "_VYG_CI_" + dateXfile + ".rtf"
  End If
End If

ActiveDocument.SaveAs2 FileName:=str2save, _
FileFormat:=wdFormatRTF
str2save = wPath + "\Resultados\Tables.doc"
ActiveDocument.SaveAs2 FileName:=str2save, _
FileFormat:=wdFormatDOC

oExcel.Workbooks.Close

dataXfile = Creador_tablas

Creador_tablas = wPath

End Function
```

5.1.5. Table of Contents

El *Table of Contents* es un documento que traza un bus de la aeronave con la página o páginas donde se encuentra su información dentro del documento *Tables*. Por ello, pese a que al cliente en su publicación le aparecerá antes este documento que el de *Tables*, es necesario que ejecutemos primero la macro que genera este segundo archivo. Puede considerarse como un índice de la tabla de información técnica del manual.

Puesto que consta de una única tabla, emplearemos una plantilla base y nos limitaremos a ir rellenando la información a medida que vayamos leyendo el archivo *Tables*. Cuando toda la información haya sido incluida, si el número de páginas es impar, añadiremos una última página en blanco con el texto *"This page left blank intentionally"*.

La plantilla debe tener en la cabecera el logo de la empresa y en el pie de página la misma información que el *Record of Revision*.

El código es el siguiente:

```
Function Creador_TOC(rev As String, MSN As String, FSN As String,_
_ fleet As String, CoM As String, wPath As String)

'Abrimos el arhcivo Tablas asociado
Dim tabPath As String
tabPath = wPath + "\Resultados\Tables.doc"
Documents.Open (tabPath)

'Extraemos la información de Tablas y la metemos en la matriz MatTab
ReDim MatTab(Documents("Tables.doc").Tables.Count, 3)
Dim i As Integer
Dim k As Integer
i = 1
k = 1
While i < Documents("Tables.doc").Tables.Count
MatTab(k, 1) = Left(Documents("Tables.doc").Tables(i)._
_Cell(i, 2).Range, Len(Documents("Tables.doc").Tables(i)._
_Cell(i, 2).Range) - 2)

  MatTab(k, 2) = Left(Documents("Tables.doc").Tables(i)._
  _Cell(i, 3).Range, Len(Documents("Tables.doc").Tables(i)._
  _Cell(i, 3).Range) - 2)
  MatTab(k, 3) = Documents("Tables.doc").Tables(i)._
  _Range.Information(wdActiveEndPageNumber)
  i = i + 2
  k = k + 1
Wend

'Abrimos el archivo de Resultados donde vamos a cargar la información
Dim tocPath As String
tocPath = wPath + "\Plantillas\04_Table of Contents.doc"
Documents.Open (tocPath)

'Cargamos información
Dim j As Integer
j = 1
While j < k
  If j < Documents("04_Table of Contents.doc").Tables(1)._
  _Rows.Count Then

    Documents("04_Table of Contents.doc").Tables(1).Cell(j + 1, 1)._
    _Range = "SUB-BUSBAR"
    Documents("04_Table of Contents.doc").Tables(1).Cell(j + 1, 2)._
    _Range = MatTab(j, 1) & " - " & MatTab(j, 2)
    Documents("04_Table of Contents.doc").Tables(1).Cell(j + 1, 4)._
```

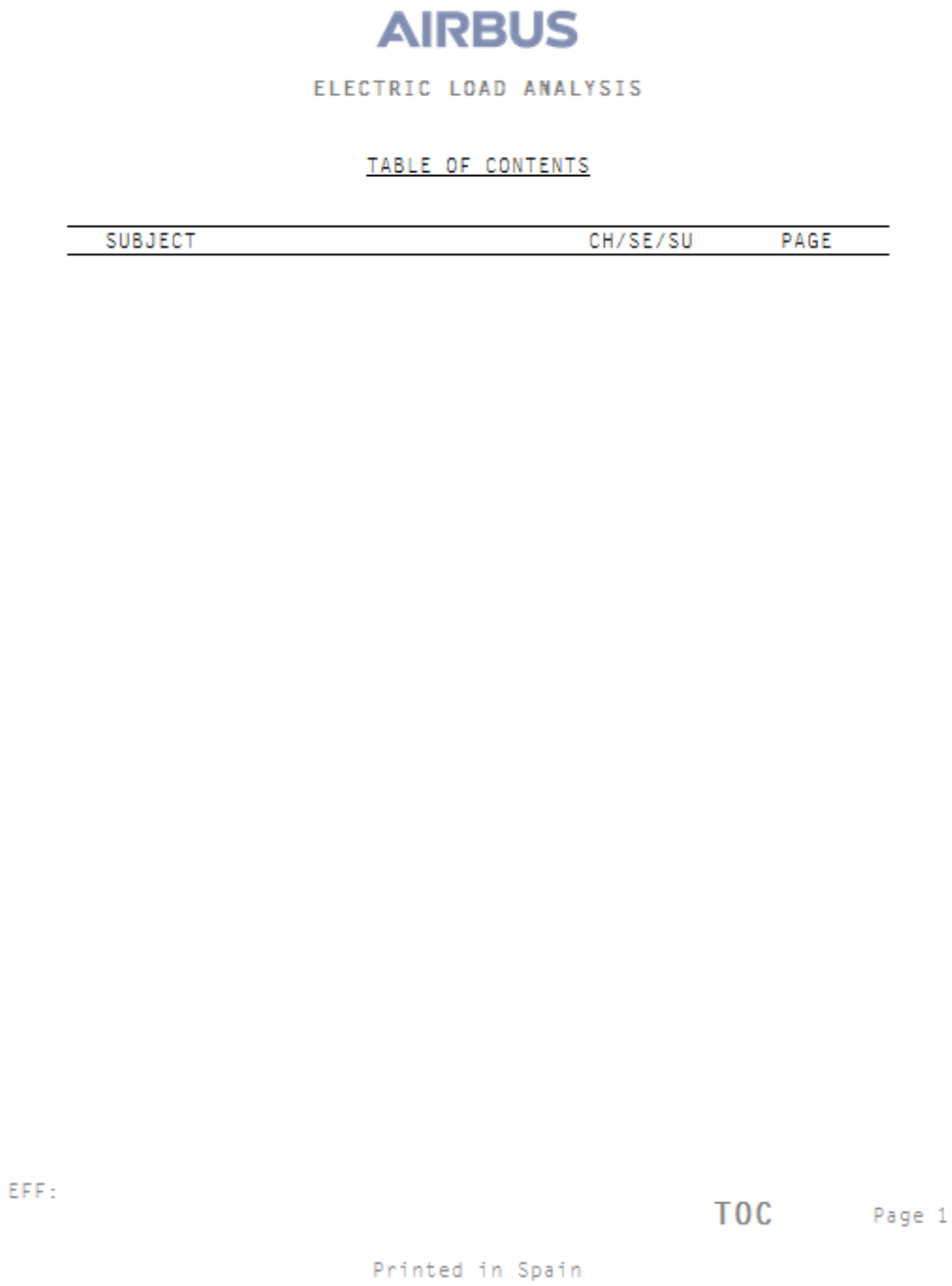


Figura 5.4: Plantilla de Table of Contents

```
    _Range = MatTab(j, 3)
  Else
    Documents("04_Table of Contents.doc").Tables(1).Rows.Add
    Documents("04_Table of Contents.doc").Tables(1).Cell(j + 1, 1).
    _Range = "SUB-BUSBAR"
    Documents("04_Table of Contents.doc").Tables(1).Cell(j + 1, 2).
    _Range = MatTab(j, 1) & " - " & MatTab(j, 2)
    Documents("04_Table of Contents.doc").Tables(1).Cell(j + 1, 4).
    _Range = MatTab(j, 3)
  End If
  j = j + 1
Wend

'Rellenamos tabla con pasos intermedios

j = 3
While j <= Documents("04_Table of Contents.doc").Tables(1).Rows.Count
  Dim preValue
  preValue = Left(Documents("04_Table of Contents.doc").Tables(1).
  _Cell(j - 1, 4).Range, Len(Documents("04_Table of Contents.doc").
  _Tables(1).Cell(j - 1, 4).Range) - 1)

  Dim numPreValue As Integer
  numPreValue = CInt(preValue) + 1

  If Left(Documents("04_Table of Contents.doc").Tables(1).
  _Cell(j, 4).Range, Len(Documents("04_Table of Contents.doc").
  _Tables(1).Cell(j, 4).Range) - 1) = numPreValue Then

    j = j + 1
  Else
    Documents("04_Table of Contents.doc").Tables(1).Rows.Add_
    _BeforeRow:=Documents("04_Table of Contents.doc").
    _Tables(1).Rows(j)
    Documents("04_Table of Contents.doc").Tables(1).Cell(j, 1).Range_
    _ = Documents("04_Table of Contents.doc").Tables(1).
    _Cell(j - 1, 1).Range
    Documents("04_Table of Contents.doc").Tables(1).Cell(j, 2).Range_
    _ = Documents("04_Table of Contents.doc").Tables(1).
    _Cell(j - 1, 2).Range
    Documents("04_Table of Contents.doc").Tables(1).Cell(j, 4).Range_
    _ = CInt(Left(Documents("04_Table of Contents.doc").Tables(1).
    _Cell(j - 1, 4).Range, Len(Documents("04_Table of Contents.doc").
    _Tables(1).Cell(j - 1, 4).Range) - 1)) + 1
  End If
Wend

'Si el número de páginas es impar, añadimos una en  blanco
If Documents("04_Table of Contents.doc").ComputeStatistics_
_(wdStatisticPages) Mod 2 <> 0 Then

  Set myRange = Documents("04_Table of Contents.doc")._Paragraphs_
  _(Documents("04_Table of Contents.doc").Paragraphs.Count).Range
```

```
  With myRange
    .Collapse Direction:=wdCollapseEnd
    .InsertBreak Type:=wdPageBreak
  End With
  Documents("04_Table of Contents.doc").Paragraphs_
  _(Documents("04_Table of Contents.doc").Paragraphs.Count)._
  _Range.Collapse Direction:=wdCollapseEnd
  Documents("04_Table of Contents.doc").Paragraphs(Documents_
  _("04_Table of Contents.doc").Paragraphs.Count).Range.InsertBreak_
  _Type:=wdSectionBreakContinuous

  Documents("04_Table of Contents.doc").Paragraphs_
  _(Documents("04_Table of Contents.doc").Paragraphs.Count)._
  _Range.InsertAfter "This page left blank intentionally"

  Documents("04_Table of Contents.doc").Sections(2).PageSetup._
  _VerticalAlignment = wdAlignVerticalCenter
  Documents("04_Table of Contents.doc").Paragraphs_
  _(Documents("04_Table of Contents.doc").Paragraphs.Count).Range._
  _ParagraphFormat.Alignment = wdAlignParagraphCenter
 End If

 'Rellenamos pie de página
 Dim str_mfsn As String
 If CoM = "" Then
   str_mfsn = "FSN " & FSN & " (MSN " & MSN & ")"
 Else
   If CoM = "MILITARY" Then
   str_mfsn = "FSN " & FSN & " (MSN " & MSN & ") VYG-KC"
 Else
   str_mfsn = "FSN " & FSN & " (MSN " & MSN & ") VYG-CI"
 End If
End If
Documents("04_Table of Contents.doc").Sections(1).Footers_
_(wdHeaderFooterPrimary).Range.Tables(1).Range.Cells(2).Range._
_InsertAfter str_mfsn
Documents("04_Table of Contents.doc").Sections(1).Footers_
_(wdHeaderFooterPrimary).Range.Tables(1).Range.Cells(6).Range._
_InsertAfter rev

'Guardamos
Dim finalPath As String
finalPath = wPath + "\Resultados\01-Manual Front Matter_
_\04_Table of Contents.doc"
Documents("04_Table of Contents.doc").SaveAs2 FileName:=finalPath, _
FileFormat:=wdFormatDOC

End Function
```

5.1.6. Introducción

Siguiendo la filosofía de las anteriores funciones, la Introducción es un documento escrito con imágenes de contenido idéntico para todos los aviones que explica en qué con-

siste el manual, cómo se distribuye la información en el mismo y cómo debe utilizarse para poder interpretar correctamente la información.

La macro asociada únicamente deberá cambiar sobre la plantilla la fecha de revisión, MSN de la aeronave y FSN de la misma, tal y como se ha estado haciendo en documentos anteriores.

5.1.7. List of Effective Pages

Este documento del manual contiene una tabla con 4 columnas: en la primera de ellas introducimos la unidad documental que queremos identificar; en la segunda se dice si cada página de cada una de esas unidades documentales ha sido revisada, incluido como nueva o eliminada; en la tercera se añade el número de página identificativa; y, por último, en la cuarta columna se añade la fecha de revisión. En el caso del ELA, se editan todos los documentos cada vez que se publica una nueva revisión, por lo que esta fecha siempre va a coincidir con la de publicación del manual. Puede entenderse que no tiene mucho sentido este campo, pero se mantiene por consistencia con el resto de manuales editados en Word donde sí que esta fecha varía por no tener que editarse todas las unidades documentales cada revisión.

Para poder rellenar el contenido de esta tabla se parte de su plantilla y, a continuación, se abren todos los archivos de la revisión anterior y los de la nueva revisión ya generados y se comparan los números de páginas de cada documento. De este modo, si el documento *DOC* de la revisión anterior tiene n páginas, y el *DOC* de la actual revisión tiene m páginas:

- $m > n$: Tendremos marca R hasta la página n y desde la página $n+1$ hasta la página m tendremos marca N.
- $m < n$: Tendremos marca R hasta la página m y desde la página $m+1$ hasta la página n tendremos marca D.
- $m = n$: Tendremos marca R en todas las páginas.

donde $R = Revised$, $N = New$ y $D = Deleted$.

El código a implementar será el siguiente:

```
Function Creador_LEP(rev As String, MSN As String, FSN As String, _
_fleet As String, CoM As String, wPath As String)

'Abrimos el documento sobre el que vamos a trabajar
Dim nPath As String
nPath = wPath + "\Plantillas\03_List of Effective Pages_p.doc"
Documents.Open (nPath)

'Cerramos los documentos que tenemos abierto y no nos sirven
Documents("04_Table of Contents.doc").Close

'Creamos las variables con las que vamos a jugar
Dim TOCold As String
Dim TOCnew As String
Dim INTROold As String
Dim INTROnew As String
```

AIRBUS

ELECTRIC LOAD ANALYSIS

LIST OF EFFECTIVE PAGES

N, R or D indicate pages which are New, Revised or Deleted respectively.
Remove and insert the affected pages and complete the Record of the Revisions.

CH/SE/SU	MVT	PAGE	ISSUE DATE	CH/SE/SU	MVT	PAGE	ISSUE DATE

EFF:

LEP

Page 1

Printed in Spain

Figura 5.5: Plantilla de List of Effective Pages

```
Dim TABLESold As String
Dim TABLESnew As String

'Cargamos el valor de las variables
Dim auxPath As String

''Datos de revisión anterior
auxPath = wPath + "\MFM anterior\02_Record of Revisions.doc"
Documents.Open (auxPath)
TOCold = Documents("02_Record of Revisions.doc").
_ComputeStatistics(wdStatisticPages)
Documents("02_Record of Revisions.doc").Close_
_ SaveChanges:=wdDoNotSaveChanges

auxPath = wPath + "\MFM anterior\05_Introduction.doc"
Documents.Open (auxPath)
INTROold = Documents("05_Introduction.doc").
_ComputeStatistics(wdStatisticPages)
Documents("05_Introduction.doc").Close_
_ SaveChanges:=wdDoNotSaveChanges

auxPath = wPath + "\MFM anterior\Tables_old.doc"
Documents.Open (auxPath)
TABLESold = Documents("Tables_old.doc").
_ComputeStatistics(wdStatisticPages)
Documents("Tables_old.doc").Close_
_SaveChanges:=wdDoNotSaveChanges

''Datos de nueva revisión
auxPath = wPath + "\Resultados\01-Manual Front Matter_
_\02_Record of Revisions.doc"
Documents.Open (auxPath)
TOCnew = Documents("02_Record of Revisions.doc").
_ComputeStatistics(wdStatisticPages)
Documents("02_Record of Revisions.doc").Close_
_SaveChanges:=wdDoNotSaveChanges

auxPath = wPath + "\Resultados\01-Manual Front Matter_
_\05_Introduction.doc"
Documents.Open (auxPath)
INTROnew = Documents("05_Introduction.doc").
_ComputeStatistics(wdStatisticPages)
Documents("05_Introduction.doc").Close SaveChanges_
_:=wdDoNotSaveChanges

TABLESnew = Documents("Tables.doc").
_ComputeStatistics(wdStatisticPages)

'Rellenamos la tabla
''Los datos de la ROR serán constantes
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(2, 1).Range = "ROR"
Documents("03_List of Effective Pages_p.doc").
```

```
_Tables(1).Cell(3, 1).Range = "ROR"
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(2, 2).Range = "R"
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(3, 2).Range = "R"
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(2, 3).Range = "1"
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(3, 3).Range = "2"
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(2, 4).Range = rev
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(3, 4).Range = rev

''Dejamos fila en blanco
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(4, 1).Range = ""
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(4, 2).Range = ""
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(4, 3).Range = ""
Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(4, 4).Range = ""

''Dejamos los datos relativos a la LEP para el final
''Rellenamos datos relativos al TOC
Dim tc As Integer '( tc = Table Counter)
Dim tocC As Integer '( tocC = TOC Counter)
tc = 5
tocC = 1

While tocC <= TOCold Or tocC <= TOCnew
  Documents("03_List of Effective Pages_p.doc")._
  _Tables(1).Rows.Add
  Documents("03_List of Effective Pages_p.doc")._
  _Tables(1).Cell(tc, 1).Range = "TOC"
  If tocC <= TOCold And tocC > TOCnew Then
    Documents("03_List of Effective Pages_p.doc")._
    Tables(1).Cell(tc, 2).Range = "D"
  ElseIf tocC > TOCold And tocC <= TOCnew Then
    Documents("03_List of Effective Pages_p.doc")._
    _Tables(1).Cell(tc, 2).Range = "N"
  Else
    Documents("03_List of Effective Pages_p.doc")._
    _Tables(1).Cell(tc, 2).Range = "R"
  End If
  Documents("03_List of Effective Pages_p.doc")._
  _Tables(1).Cell(tc, 3).Range = tocC
  Documents("03_List of Effective Pages_p.doc")._
  _Tables(1).Cell(tc, 4).Range = rev
  tc = tc + 1
  tocC = tocC + 1
Wend
```

```
''Dejamos fila en blanco
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Rows.Add
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(tc, 1).Range = ""
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(tc, 2).Range = ""
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(tc, 3).Range = ""
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(tc, 4).Range = ""
tc = tc + 1

''Rellenamos datos relativos a la INTRO
Dim intC As Integer '( intC = INTRO Counter)
intC = 1
While intC <= INTROold Or intC <= INTROnew
  Documents("03_List of Effective Pages_p.doc").
  _Tables(1).Rows.Add
  Documents("03_List of Effective Pages_p.doc").
  _Tables(1).Cell(tc, 1).Range = "INTRO"
  If intC <= INTROold And intC > INTROnew Then
    Documents("03_List of Effective Pages_p.doc").
    _Tables(1).Cell(tc, 2).Range = "D"
  ElseIf intC > INTROold And intC <= INTROnew Then
    Documents("03_List of Effective Pages_p.doc").
    _Tables(1).Cell(tc, 2).Range = "N"
  Else
    Documents("03_List of Effective Pages_p.doc").
    _Tables(1).Cell(tc, 2).Range = "R"
  End If
  Documents("03_List of Effective Pages_p.doc").
  _Tables(1).Cell(tc, 3).Range = intC
  Documents("03_List of Effective Pages_p.doc").
  _Tables(1).Cell(tc, 4).Range = rev
  tc = tc + 1
  intC = intC + 1
Wend

''Dejamos fila en blanco
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Rows.Add
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(tc, 1).Range = ""
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(tc, 2).Range = ""
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(tc, 3).Range = ""
Documents("03_List of Effective Pages_p.doc").
_Tables(1).Cell(tc, 4).Range = ""
tc = tc + 1
```

```
''Rellenamos datos relativos a las TABLAS
Dim tabC As Integer '( tabC = TABLES Counter)
While tabC <= TABLESold Or tabC <= TABLESnew
  Documents("03_List of Effective Pages_p.doc").
  _Tables(1).Rows.Add
  Documents("03_List of Effective Pages_p.doc").
  _Tables(1).Cell(tc, 1).Range = "ELA"
  If tabC <= TABLESold And tabC > TABLESnew Then
    Documents("03_List of Effective Pages_p.doc").
    _Tables(1).Cell(tc, 2).Range = "D"
  ElseIf tabC > TABLESold And tabC <= TABLESnew Then
    Documents("03_List of Effective Pages_p.doc").
    _Tables(1).Cell(tc, 2).Range = "N"
  Else
    Documents("03_List of Effective Pages_p.doc").
    _Tables(1).Cell(tc, 2).Range = "R"
  End If
  Documents("03_List of Effective Pages_p.doc").
  _Tables(1).Cell(tc, 3).Range = tabC + 1
  Documents("03_List of Effective Pages_p.doc").
 _Tables(1).Cell(tc, 4).Range = rev
 tc = tc + 1
 tabC = tabC + 1
Wend

'Rellenamos pie de página
Dim str_mfsn As String
If CoM = "" Then
  str_mfsn = "FSN " & FSN & " (MSN " & MSN & ")"
  Else
  If CoM = "MILITARY" Then
    str_mfsn = "FSN " & FSN & " (MSN " & MSN & ") VYG-KC"
  Else
    str_mfsn = "FSN " & FSN & " (MSN " & MSN & ") VYG-CI"
  End If
End If
Documents("03_List of Effective Pages_p.doc").Sections(1).
_Footers(wdHeaderFooterPrimary).Range.Tables(1).Range.
_Cells(2).Range.InsertAfter str_mfsn
Documents("03_List of Effective Pages_p.doc").Sections(1).
_Footers(wdHeaderFooterPrimary).Range.Tables(1).Range.
_Cells(6).Range.InsertAfter rev

''Rellenamos columnas de LEP
Dim LEPold As String
auxPath = wPath + "\MFM anterior\03_List of Effective Pages.doc"
Documents.Open (auxPath)
LEPold = Documents("03_List of Effective Pages.doc").
_ComputeStatistics(wdStatisticPages)
Documents("03_List of Effective Pages.doc").Close_
_ SaveChanges:=wdDoNotSaveChanges

Dim LEPnew As String
```

```
LEPnew = Documents("03_List of Effective Pages_p.doc")._
_ComputeStatistics(wdStatisticPages)

Dim lepC As Integer '( lepC = LEP Counter)
lepC = 1
While lepC <= LEPold Or lepC <= LEPnew
  Documents("03_List of Effective Pages_p.doc").Tables(1).Rows.Add_
  _ BeforeRow:=Documents("03_List of Effective Pages_p.doc")._
  _Tables(1).Rows(4 + lepC)
  Documents("03_List of Effective Pages_p.doc").Tables(1)._
  _Cell(4 + lepC, 1).Range = "LEP"
  Documents("03_List of Effective Pages_p.doc").Tables(1)._
  _Cell(4 + lepC, 3).Range = lepC
  Documents("03_List of Effective Pages_p.doc").Tables(1)._
  _Cell(4 + lepC, 4).Range = rev
  lepC = lepC + 1
  LEPnew = Documents("03_List of Effective Pages_p.doc")._
  _ComputeStatistics(wdStatisticPages)
Wend

''Añadimos una página en blanco si no es par
If Documents("03_List of Effective Pages_p.doc")._
_ComputeStatistics(wdStatisticPages) Mod 2 <> 0 Then
  If LEPold < LEPnew Then
    Documents("03_List of Effective Pages_p.doc").Tables(1).Rows.Add_
    _ BeforeRow:=Documents("03_List of Effective Pages_p.doc")._
    _Tables(1).Rows(4 + lepC)
    Documents("03_List of Effective Pages_p.doc").Tables(1)._
    _Cell(4 + lepC, 1).Range = "LEP"
    Documents("03_List of Effective Pages_p.doc").Tables(1)._
    _Cell(4 + lepC, 3).Range = lepC
    Documents("03_List of Effective Pages_p.doc").Tables(1)._
    _Cell(4 + lepC, 4).Range = rev
  End If

  Set myRange = Documents("03_List of Effective Pages_p.doc")._
 _Paragraphs(Documents("03_List of Effective Pages_p.doc")._
   _Paragraphs.Count).Range
  With myRange
    .Collapse Direction:=wdCollapseEnd
    .InsertBreak Type:=wdPageBreak
  End With
  Documents("03_List of Effective Pages_p.doc").Paragraphs_
  _(Documents("03_List of Effective Pages_p.doc")._
  _Paragraphs.Count).Range.Collapse Direction:=wdCollapseEnd
  Documents("03_List of Effective Pages_p.doc").Paragraphs_
  _(Documents("03_List of Effective Pages_p.doc").Paragraphs._
  _Count).Range.InsertBreak Type:=wdSectionBreakContinuous

  Documents("03_List of Effective Pages_p.doc").Paragraphs_
  _(Documents("03_List of Effective Pages_p.doc").Paragraphs._
  _Count).Range.InsertAfter "This page left blank intentionally"
```

```
  Documents("03_List of Effective Pages_p.doc").Paragraphs_
  _(Documents("03_List of Effective Pages_p.doc").Paragraphs.Count)._
  _Range.PageSetup.VerticalAlignment = wdAlignVerticalCenter
  Documents("03_List of Effective Pages_p.doc").Paragraphs_
  _(Documents("03_List of Effective Pages_p.doc").Paragraphs.Count)._
  _Range.ParagraphFormat.Alignment = wdAlignParagraphCenter

  lepC = lepC - 1

End If

Documents("03_List of Effective Pages_p.doc").Tables(1).Rows.Add_
_ BeforeRow:=Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Rows(5 + lepC)

''Rellenamos N, D o R según corresponda en la tabla
LEPnew = Documents("03_List of Effective Pages_p.doc")._
_ComputeStatistics(wdStatisticPages)
Dim lastCounter As Integer
lastCounter = 5
While Left(Documents("03_List of Effective Pages_p.doc")._
_Tables(1).Cell(lastCounter, 1).Range, 3) = "LEP"
  If lastCounter - 4 <= LEPold And lastCounter - 4 <= LEPnew Then
    Documents("03_List of Effective Pages_p.doc").Tables(1)._
    _Cell(lastCounter, 2).Range = "R"
  ElseIf lastCounter - 4 <= LEPold And lastCounter - 4 > LEPnew Then
    Documents("03_List of Effective Pages_p.doc").Tables(1)._
    _Cell(lastCounter, 2).Range = "D"
  Else
    Documents("03_List of Effective Pages_p.doc").Tables(1)._
    _Cell(lastCounter, 2).Range = "N"
  End If
lastCounter = lastCounter + 1
Wend

'Guardamos
Dim finalPath As String
finalPath = wPath + "\Resultados\01-Manual Front Matter\_
_03_List of Effective Pages.doc"
Documents("03_List of Effective Pages_p.doc").SaveAs2_
_FileName:=finalPath, FileFormat:=wdFormatDOC

End Function
```

6. Conclusiones

A la hora de automatizar una tarea uno puede encontrarse varias dificultades para poder llevar a cabo su objetivo satisfactoriamente. En primer lugar, será necesario elegir el software y lenguaje de programación más adecuados para poder hacer la tarea y no renunciar a aprender un nuevo lenguaje por ya conocer uno con el que puede llevarse a cabo la misma tarea: se necesitará un tiempo de formación previo que implicará un retraso en las primeras etapas del trabajo pero, si es el adecuado, se acabará realizando las tareas no sólo antes, sino de manera más sencilla y, por tanto, más robusta.

En la concepción de este TFG se propuso que las automatizaciones fuesen totales y no fuese necesaria la mano de ninguna persona para acciones intermedias. Debido a su complejidad, se acabó tomando la determinación de que quedase fuera del alcance del trabajo. Para conectar softwares independientes como lo son el FOXE y Excel, hubiera sido necesario programar DLLs (Dynamic Library Links) [9], de modo que se hubiera convertido el lenguaje de FOXE a *C* para poder haber realizado llamadas desde VBA y que el mismo Excel hubiera sido quien hubiese ejecutado las acciones del FOXE.

Tal y como se ha comentado durante este trabajo, la figura del Clean-up ha surgido recientemente. Se trata de un nuevo campo a explorar y desarrollar sin precendentes para el programa y la empresa tratados en este trabajo. La idea original para esta figura era la de un trabajador poco cualificado que de manera manual fuese capaz de detectar errores empleando una gran cantidad de tiempo y llevando a cabo labores repetitivas de manera periódica. No obstante, la realización de este TFG me ha permitido aprender en profundidad la programación necesaria para poder automatizar estas tareas, convirtiéndose dichas automatizaciones actualmente en piezas imprescindibles para el programa en fase de implementación y permitiéndome un salto considerable en mi valoración como profesional.

7. Trabajos futuros

Este Trabajo de Fin Grado ha permitido demostrar que la correcta programación permite la extracción de toda información que se encuentre en un documento desarrollado bajo un lenguaje de marcado y la clasificación de ésta de manera ordenada.

El objetivo del Clean-up debe ser la obtención automática de manera periódica de los errores clásicos o típicos en las PPTT, además de ser capaz de trazar cualquier nuevo caso. Para ello, el siguiente paso que debe darse, y con ello que el proyecto alcance un estado de madurez profesional, es la organización en bases de datos de toda la información clasificable de todos los manuales. Para ello, al inicio de cada revisión se ejecutarían todas las macros que extraigan esta información y se organizarían las bases de datos que se comentan. Así, cada vez que sea necesario obtener una información determinada, sólo deberán hacerse extracciones sobre éstas: mucho más sencillo y rápido.

Esta información quedaría perfectamente recogida y guardada revisión tras revisión y serviría como backup: podría analizarse con facilidad cualquier evolución temporal de las PPTT.

Si se pudiese implementar esta idea, con un pequeño curso de formación sobre cruces con bases de datos podría enseñarse a cualquier trabajador sin formación en programación a acceder en todo momento a la información que desee de manera independiente. Esto permitirá hacer el trabajo más eficiente y minuicioso: los validadores tendrán trazadas las tareas en las que hay que implementar determinados cambios gracias a los cruces que puedan hacer con las bases de datos y no tendrán que analizar el documento que deban validar para saber si hay que implementar dicho cambio. Con ello se podrá ahorrar tiempo y reducir las posibilidades de error.

Con frecuencia, los trabajadores de un manual determinado proponen el análisis de posibles errores que no se llega a ejecutar porque carecen de perfiles capaces de realizar las extracciones en sus empresas. Con estas bases de datos, en el perfeccionamiento de las publicaciones podrá participar cualquier trabajador: tendrán una herramienta a su disposición que les permitirá hacer el análisis que se les ocurra.

Otro campo a explorar por la figura del Clean-up debería ser la conversión de los manuales desarrollados con Word a lenguaje de etiquetas, de modo que estos manuales también puedan participar en la idea que se propone de trabajo futuro.

Referencias

[1] Airbus Company. *https://www.airbus.com/company/we-are-airbus.html*.

[2] Airbus A330-MRTT. *https://www.airbus.com/defence/a330mrtt.html*.

[3] Winglet definition. *https://www.nasa.gov/centers/dryden/about/Organizations_/Technology/Facts/TF-2004-15-DFRC.html*.

[4] Arbortext Editor. *https://gpsl.co/product/arbortext-editor*.

[5] DTD definition. *https://www.w3schools.com/xml/xml_dtd_intro.asp*.

[6] BaseX. *http://basex.org/basex/*.

[7] First Object XML Editor. *http://www.firstobject.com/index.html*.

[8] XQuery Language. *https://www.w3schools.com/xml/xquery_intro.asp*.

[9] DLL. *https://support.microsoft.com/es-es/help/815065/what-is-a-dll*.