UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

OPTIMIZACIÓN DE UN SISTEMA PARA LA DISEMINACIÓN DE MODELOS PLM EN UNA EMPRESA DE DESARROLLO DE VEHÍCULOS FERROVIARIOS

AUTORA: Isabel Lázaro Miguel-Sin

TUTOR: Joaquín Gracia Morán

Curso Académico: 2016-17

Agradecimientos

Me gustaría agradecer a la empresa Stadler Rail S.A.U., la oportunidad de realizar un proyecto como este, donde he podido aplicar mis conocimientos de forma práctica en casos reales como el del presente TFG. En concreto, agradecer a mi tutor de la empresa, Guillermo Bruixola Casañ, su gran profesionalidad y cercana dedicación.

También quiero agradecer a mi tutor de la Universidad Politécnica de Valencia la paciencia que ha tenido conmigo en la realización del trabajo y su dedicación y apoyo constantes.

A compañeros de universidad, que me han apoyado y ayudado en la elaboración de este trabajo.

Y finalmente, agradecer a mi familia la alegría y descanso que es llegar a casa, un sitio donde siempre hay alguien dispuesto a ayudar y que apoyan cualquier decisión que se tome en la vida. Sin vosotros nada hubiese sido posible.

Resumen

En este Trabajo Final de Grado se ha diseñado e implementado un sistema para la diseminación automática de los modelos desarrollados en el entorno PLM, de tal manera que se utilizan aquellas horas de la jornada laboral en las que el entorno no es usado. De esta forma, la actualización de dichos modelos no perturba la jornada laboral de los ingenieros.

Para realizar este TFG, se han desarrollado una serie de módulos que analizan las variaciones realizadas en los diferentes ensamblajes PLM, y aquellas novedades detectadas, se diseminan por el resto de componentes.

Así mismo, se han implementado una serie de módulos para realizar diferentes tareas que pueden llevarse a cabo de forma automática, aprovechando parte de la aplicación de diseminación de modificaciones.

Palabras clave: software PLM, entorno del sistema, entorno de diseño, actualización de estructura, eficiencia, Siemens, Stadler Rail.

Resum

En aquest Treball Final de Grau s'ha dissenyat e implementat un sistema per a la disseminació automàtica dels models desenvolupats en l'entorn PLM, de tal manera que s'utilitzen aquelles hores de la jornada laboral en què l'entorn no és usat. D'aquesta manera, l'actualització d'aquests models no pertorba la jornada laboral dels enginyers.

Per realitzar aquest TFG, s'han desenvolupat una sèrie de mòduls que analitzen les variacions realitzades en els diferents acoblaments PLM, i aquelles novetats detectades, es disseminen per la resta de components.

Així mateix, s'han implementat una sèrie de mòduls per realitzar diferents tasques que es poden dur a terme de forma automàtica, aprofitant part de l'aplicació de disseminació de modificacions.

Paraules clau: software PLM, entorn del sistema, entorn de disseny, actualització d'estructura, eficiència, Siemens, Stadler Rail.

Abstract

In this Thesis, a system for the automatic dissemination of the models developed in the PLM environment has been designed and implemented, so that the hours of the workday in which the environment is not used are the ones that this application uses. In this way, the updating of these models does not disturb the working day of the engineers.

In order to realize this TFG, a series of modules, that have been developed, analyze the variations made in the different PLM assemblies, and those innovations detected, are disseminated by the rest of components.

Also, a series of modules have been implemented to perform different tasks that can be carried out automatically, taking advantage of part of the application that disseminates the modifications.

Keywords: PLM software, system environment, design environment, structure update, efficiency, Siemens, Stadler Rail.

INDICE

MEMORIA DESCRIPTIVA

Capítulo 1. **INTRODUCCIÓN**

1.1 Antecedentes del proyecto

En el presente TFG se ha realizado en la planta valenciana de *Stadler Rail*, que se encuentra en el polígono industrial de Albuixec, a diez minutos de Valencia. Es una compañía internacional del sector ferroviario especializada en la venta de locomotoras y trenes ligeros entre otros productos. Esta planta remonta sus orígenes a un taller valenciano llamado Talleres Devis, que fue creciendo hasta formar parte de la empresa francesa *Alstom*, más tarde de la empresa alemana *Vossloh* y en la actualidad de la compañía suiza *Stadler Rail S.A.U*. En concreto, se va a mejorar parte del software PLM que utiliza esta empresa.

En la compañía, y en concreto, en el departamento en el que se ha desarrollado este TFG se realizan multitud de labores diferentes, todas necesarias. Sin embargo, no todas precisan de una supervisión en todo momento, si no que podrían realizarse de forma automática y de esta manera, se podría emplear ese tiempo en realizar el resto de tareas. De ahí surge la propuesta de generar una herramienta que ejecute estas tareas fuera del horario laboral, por la noche, y de este modo, centrar las horas de jornada laboral en realizar tareas que se deben realizar sin automatizarlas.

Una de esas tareas, y en la que se centra el presente TFG, es la actualización de los componentes en la estructura del producto. La diseminación de la información referente al producto que se fabrica conviene que se realice de forma rápida y dinámica para no comprometer la efectividad de la compañía. Para ello, el software de diseño permite acceder y editar esa información desde cualquier punto mediante cuentas de usuario de los trabajadores. La información, por lo tanto, está sometida a constantes cambios en diferentes localizaciones, y en ocasiones las modificaciones de los componentes no se actualizan en la estructura del producto.

Hasta el momento, la actualización de la información delos componentes modificados se realizaba mediante una aplicación facilitada por el propio PLM. Sin embargo, dicha aplicación no estaba pensada para emplearla en un producto con más de 10.000 componentes como es una locomotora, y por ello el proceso presentaba un gran coste temporal. Además, durante ese tiempo, el ingeniero que lleva a cabo la tarea se queda parcialmente sin herramienta de trabajo al ser un proceso que precisa de una licencia de software y gran parte de la potencia de memoria RAM del ordenador. De ahí surge la propuesta de una mejora de dicha aplicación.

1.2 Objetivos del proyecto

El objetivo del presente TFG es exponer el desarrollo y la aplicabilidad de una herramienta que realiza de forma automática ciertas tareas repetitivas.

De forma resumida, los objetivos del TFG en su conjunto serían:

- Desarrollar una herramienta multifuncional que disponga de los medios necesarios para efectuar distintas tareas desde la misma aplicación.
- Implementar una aplicación que se adapte a las necesidades de actualización de un determinado proyecto y que, para cada uno, utilice la forma óptima de realizar dicha tarea.

- Mejorar la eficiencia del sistema en la actualización de datos realizando, de forma sencilla y rápida, la diseminación de las modificaciones de los componentes cambiados a través de toda la estructura del producto, mediante una herramienta que realizará un estudio sobre cómo abordar la actualización de la forma más eficiente y rápida posible.

1.3 Alcance del proyecto

Este proyecto recoge la generación de cinco aplicaciones distintas que facilitan varias tareas de gestión de datos del departamento de Ingeniería de la empresa *Stadler Rail*. Dichas aplicaciones son muy específicas, por lo que el presente trabajo se centra sobre todo en el entorno para el que han sido programadas: el entorno de diseño del producto.

La primera aplicación genera una estructura de datos necesaria para el funcionamiento del resto de aplicaciones.

La segunda aplicación realiza, de todas las piezas modificadas, una actualización de los conjuntos que tienen asociada dicha pieza, pero, que no habían realizado el cambio de ese componente al haber modificado la pieza individualmente.

En el planteamiento inicial del proyecto, el alcance de éste abarcaba sólo las actualizaciones de la estructura del producto. Sin embargo, tras ver la aplicabilidad de las herramientas generadas, se desarrollaron tres aplicaciones más basándose en las dos primeras, empleando para ello los recursos utilizados por estas dos primeras aplicaciones.

En este sentido, la tercera aplicación lleva a cabo el análisis de intersecciones entre componentes de un mismo producto o entre componentes de productos distintos.

Por último, de las dos tareas restantes, una realiza una búsqueda de los componentes que cumplen cierta característica y los lista en un fichero para, a continuación, aplicarles un proceso de mejora de calidad, y la otra genera un fichero en el que, según lo especificado por el usuario, se otorga un tipo de permiso de edición, de un componente en concreto, a una ingeniería subcontratada especificada. Ambas son aplicaciones que, aprovechando los recursos generados en la primera tarea, generan los archivos de texto necesarios para desarrollar futuras aplicaciones de mejora del producto.

En el proceso de estudio del planteamiento de las aplicaciones y en implementación de las mismas ha resultado de gran utilidad ciertos conocimientos adquiridos por el alumno a lo largo del Grado de Ingeniería en Tecnologías Industriales. Dichos conocimiento son:

- Manejo de programas de diseño para la comprensión y planteamiento de la aplicación “Actualizar Estructuras”, lo que se aprendió en asignaturas como Expresión gráfica (1º de GITI) o Ingeniería gráfica (4º de GITI).
- A resultado fundamental la base de lenguaje C y C++ adquirida en asignaturas como Informática (1º de GITI), Tecnología Electrónica (3º de GITI) o Tecnología Informática Industrial (4º de GITI) o de una forma más práctica con Matlab en Métodos Matemáticos (2º de GITI). Sin este conocimiento previo, hubiese sido prácticamente imposible que el alumno desarrollase las aplicaciones necesarias.

- La asignatura de Investigación Operativa da al alumno una visión de lo que son las tareas a realizar por una empresa, la forma de distribución del trabajo entre departamentos y la importancia de un entorno dinámico de trabajo para la gestión de los datos. Es el primer contacto del alumno con sistemas de software PLM, fundamental en el desarrollo del presente TFG.
- Por último, la asignatura de Proyectos (4º de GITI) ha facilitado la comprensión de planos de componentes por parte del alumno, así como la capacidad de realizar un presupuesto o un informe de procesos reales, llevados a cabo en la empresa, que exigían la elaboración de la justificación de las medidas tomadas al efectuar toma de decisiones. Así mismo, esta asignatura ayuda al alumno a abordar la realización del TFG, lo que conlleva hacer un análisis de viabilidad del proceso y la ejecución del mismo para, finalmente, realizar un informe real.

1.4 Estructura de descomposición del proyecto (EDP)

En los siguientes capítulos se va a explicar cómo se ha llevado a cabo la ejecución del presente TFG, así como los resultados de la aplicabilidad del mismo.

En primer lugar, el Capítulo 2 es una introducción de la historia de la aparición del software de diseño moderno al ser éste el entorno de trabajo en el que se basa este proyecto.

A continuación, en el Capítulo 3 se presenta el marco teórico que explica los conceptos básicos sobre el entorno en el que se ha implementado la mejora con la programación realizada. Dicho entorno no es otro que el software de la compañía mediante el cual se realiza todo el proceso de fabricación del producto.

En el Capítulo 4 se expone de forma breve el funcionamiento de la herramienta que necesita la mejora y se justifica la necesidad de realizar dicha mejora.

En siguiente lugar, en el Capítulo 5 se detalla cómo se han implementado todas las aplicaciones referentes a la actualización de una estructura, incluyendo diagramas de flujo que reflejan de forma gráfica el funcionamiento lógico de la aplicación (contenidos en los Anexos correspondientes).

También en el Capítulo 5 se explica cómo se ha reutilizado parte de la función de la aplicación de actualización de estructuras en otras aplicaciones que realizan tareas diferentes pero que también se pueden ejecutar de forma automática fuera del horario laboral.

Por último, en el Capítulo 6 se detallan los resultados de la aplicabilidad del proyecto y las conclusiones a las que se llegan a la vista de dichos resultados.

Capítulo 2. **MARCO TEÓRICO**

En todo el sector industrial, así como en cualquier empresa tanto grande como pequeña, la gestión de la información de los productos propios y externos pasa a través de un *Product Life-cycle Management* (PLM) software. Esto se debe a que la digitalización de datos se ha convertido en un fenómeno fundamental para el avance y la mejora del desarrollo de las compañías. (Sun & Wang, 2012)

En los orígenes de la informatización de datos, las empresas comenzaron a realizar diferentes tareas, como son el diseño de piezas, estudios de viabilidad o ensayos y simulaciones de "prototipos", entre otras. De esta manera, en vez de realizar planos de diseño del producto a mano descartando muchos bocetos previos, en el archivo digital se podía modificar cualquier componente, e imprimir un nuevo plano, lo que resultaba mucho más eficiente. Y lo mismo ocurría con los ensayos de los prototipos. Al poder realizarlos de forma virtual se ahorraba tiempo y material al no fabricar prototipos físicos. Sin embargo, estas innovaciones se realizaban de forma independiente entre ellas, por lo que la información de un mismo producto estaba dispersa entre los distintos programas que facilitaban su producción. Por lo tanto, para gestionar la información independiente de forma eficiente, fue necesario que surgiera un sistema que integrase en uno único, todos los programas que facilitasen el proceso de producción. De esta forma, surgieron los sistemas PLM. Estos sistemas facilitan el intercambio y sincronización de los datos del producto de forma automática, en un entorno de trabajo mucho más dinámico, tal y como se muestra en la Imagen 1 de forma gráfica. Esto derivó en la mejora de la calidad de la producción en las empresas y en el incremento de su eficiencia, entre otras muchas ventajas.

Imagen 1 Descripción gráfica de un Software PLM. (Chivaro Engineering Innovation, 2017)

Organizar y gestionar de forma correcta toda la información de un proyecto es una tarea complicada. El diseño se realiza mediante pequeños componentes que, juntándolos, forman el producto final. En ocasiones, la información entre los distintos componentes utilizados es diferente, siendo necesarios la constante actualización y el control de datos. Una herramienta muy utilizada en entornos industriales es el *Enterprise Resource Planning* (ERP). Este software realiza funciones complementarias al PLM y gestiona todos los datos que generan dichas funciones. Al trabajar con distintos formatos de datos, la sincronización entre el PLM y el SAP suele ser problemática. Con el fin de dar soporte a las

herramientas específicas de ingeniería y conseguir que la información del producto sea la misma en ambos entornos, surge el departamento de calidad digital y gestión del software de Ingeniería conocido como el ***E**ngineering **S**oftware **A**dministration* Group (De ahora en adelante se hará referencia a ese departamento mediante las siglas **ESA**).

Todo lo referente al diseño, fabricación, validación y homologación del producto se lleva a cabo en el entorno del PLM. Por otro lado, el ERP se encarga de calcular la cantidad de material necesario para la fabricación de la unidad de obra diseñada, teniendo en cuenta lo que ya hay en stock y ejecutando compras automáticamente para disponer de lo necesario en el momento justo.

El objetivo del ESA es asegurar la fiabilidad de la información en todas sus plataformas de gestión de la información, además de gestionar las modificaciones de las piezas, dar soporte técnico de cualquier herramienta relacionada con el software de ingeniería y buscar nuevas y mejores formas de realizar las tareas de diseño y gestión de la información del departamento. Como hemos dicho antes, puede plasmarse de forma distinta en el ERP y el PLM la información del mismo producto por error. Esto evidentemente debe evitarse. Por ello, de forma temporal hasta la completa integración de ambas plataformas, objetivo en el que se está trabajando, se realizan numerosas comprobaciones automatizadas que generan diferentes hojas de cálculo con los datos que no coinciden.

El presente TFG se ha realizado dentro del departamento de calidad digital y gestión del software de Ingeniería de la empresa *Stadler Rail* mediante una beca de colaboración. La empresa *Stadler* Rail ha puesto en marcha un proyecto en el cual se basa la realización de este trabajo. Por ello, su caso servirá como ejemplo para la completa comprensión de lo que un PLM puede llegar a abarcar. El software PLM que utilizan es el de la compañía Siemens, uno de los sistemas más empleados a nivel mundial.

Un PLM es un sistema de gestión de la información que puede integrar datos, procesos, sistemas empresariales y, finalmente, personas en el contexto de una empresa, en el sentido más amplio. Eso le permite gestionar esta información a lo largo de todo el ciclo de vida de un producto de forma eficaz y económica, desde la idea inicial, el diseño y la fabricación hasta el almacenamiento del producto y su eliminación. (SIEMENS PLM, 2017)

Es decir, un PLM es un conjunto de herramientas o programas que engloban todo el ciclo de vida del producto, permite disponer de la información generada desde cualquier herramienta, así como ampliarla o modificarla, e incluye desde el primer planteamiento en fase de oferta hasta la venta, distribución y soporte posventa del mismo.

En el mundo del ferrocarril, la compra de vehículos se lleva a cabo con la convocatoria de un concurso en el que se especifican las exigencias y peticiones del cliente (normalmente gobiernos o ayuntamientos, pero cada vez con más frecuencia operadores privados). Las empresas preparan una propuesta y la presentan al concurso, y en función de los puntos asignados a cada participante, gana el que más se ajuste a los requerimientos del comprador. Tras ser asignado el proyecto, se empieza a desarrollar el diseño específico de todos los componentes necesarios. En el siguiente paso, se comprueba que los ensamblajes cumplan la normativa correspondiente. Por último, se fabrica por completo el producto y se comprueba que cumpla toda la normativa vigente y todas las especificaciones del cliente para su validación, homologación y entrega al comprador.

En el proceso descrito en el párrafo anterior, el PLM participa en todas las fases. Las principales herramientas utilizadas en ingeniería son el PLM, el NX 10, *Doors* para gestión de requisitos, *See XP* para esquemas eléctricos, *Concerto* para gestión multiproyecto y el ERP. Siendo que en el departamento ESA sólo se gestiona el software PLM y NX, el presente TFG se centrará en dichos programas, descritos en el Capítulo 2.

A lo largo de la beca se han desarrollado varias herramientas que facilitan el análisis de las diferencias de datos entre las dos plataformas. El presente TFG está enfocado en una herramienta en concreto que se desarrolló con el fin de mantener lo más actualizada posible la información de la estructura de diseño de todo el conjunto de piezas del producto. Por lo tanto, el entorno de interés en el desarrollo de este trabajo es el NX 10. En dicho entorno, la herramienta *Update Structure* lleva a cabo la actualización de la estructura de las piezas que forman el producto. Sin embargo, es un proceso muy lento, poco efectivo y es necesario disponer de una licencia del programa NX 10. Todos esos condicionantes provocaban que cuando se quiere realizar una actualización de la estructura, se debe utilizar un puesto de trabajo y una licencia en un proceso que se realiza automáticamente, por lo que el trabajador de dicho puesto se quedaba parcialmente sin herramienta de trabajo durante el tiempo que durase la actualización, que podía llegar a tardar horas.

A raíz de este problema, se planteó la posibilidad de desarrollar una utilidad que ejecutase de forma autónoma dicha actualización fuera de las horas laborales, es decir, por la noche, y también que fuese más eficaz que el *Update Structures*. Realizando un estudio de viabilidad, se llegó a la conclusión de que dicho programa facilitaría que la información estuviese actualizada de forma rápida y sencilla, y solucionaría el problema de acaparar una licencia, pues éstas están muy solicitadas durante la jornada laboral.

Capítulo 3. **SIEMENS PLM SOFTWARE**

En este capítulo se van a describir brevemente los diferentes componentes del entorno que se ha utilizado durante el desarrollo del presente TFG.

3.1. Siemens TeamCenter

Al comienzo de la digitalización de datos se hizo necesaria la aparición de algún sistema que permitiese almacenar todos los diseños informatizados y gestionarlos de forma cómoda. Por ello, surgieron los *Product Data Management* (PDM) software. Son sistemas que almacenan los archivos generados en las herramientas de diseño que indican cómo fabricar los productos. En ellos se especifica tanto las medidas como los procesos de fabricación para confeccionarlos. Sin embargo, como se ha explicado anteriormente, hubo más procesos, como las simulaciones, que empezaron a realizarse a ordenador y el PDM no estaba capacitado para gestionar y asociar esa información al producto que hacía referencia. La herramienta capaz de almacenar toda la información de un mismo producto desde su concepción hasta su distribución eran los PLM. (PDM software or PLM software: what's the difference?, 2017)

De todas formas, un PLM está compuesto mayoritariamente por un sistema PDM, una herramienta CAD y cualquier otra herramienta necesaria para la producción de cada compañía. En el caso de este trabajo, *TeamCenter* realiza la función tanto de PDM como de PLM ya que es la herramienta que se encarga de gestionar y organizar la información de diseño, se accede a su entorno desde prácticamente todos los puestos de trabajo para que todos los empleados dispongan de los datos del producto en tiempo real y los actualicen simultáneamente, permite generar múltiples hojas de cálculo con distintos cruces de datos para la toma de decisiones y mucho más. (Experto PLM Siemens, 2017)

En pocas palabras, el PDM es un sistema de gestión de archivos de diseño. El PLM es una base de datos que gestiona procesos y productos. *TeamCenter* realiza ambas funciones asociando todos los datos de ambas. (PDM software or PLM software: what's the difference?, 2017)

3.2. Software NX 10

NX es un *Computer-Aided technologies* (CAx), lo que significa que es una herramienta que facilita ciertas tareas gracias al empleo de un ordenador. Estas tareas van desde el diseño, *Computer-Aided Design* (CAD), hasta la fabricación asistida por ordenador, *Computer-Aided Manufacturing* (CAM) o simulaciones virtuales de cálculo de estados de carga, *Computer-Aided Engineering* (CAE). (Chen, Zhu, & Ye, 2012)

NX es un programa que facilita la realización de modelos o componentes de forma digital y que contiene diferentes módulos que amplían sus posibilidades. En un principio, un CAD simplemente permitía diseñar planos, si era un CAD 2D, o modelos, si era un CAD 3D. Sin embargo, con la evolución del software hacía la informatización de todos los procesos, esta herramienta desarrolló módulos para la realización de más tareas sobre el modelo diseñado, y de esta forma, realizarlas a ordenador y no de forma experimental.

3.2.1. Herramientas NX 10

En la Figura 1 se esquematizan las posibles tareas a realizar en NX.

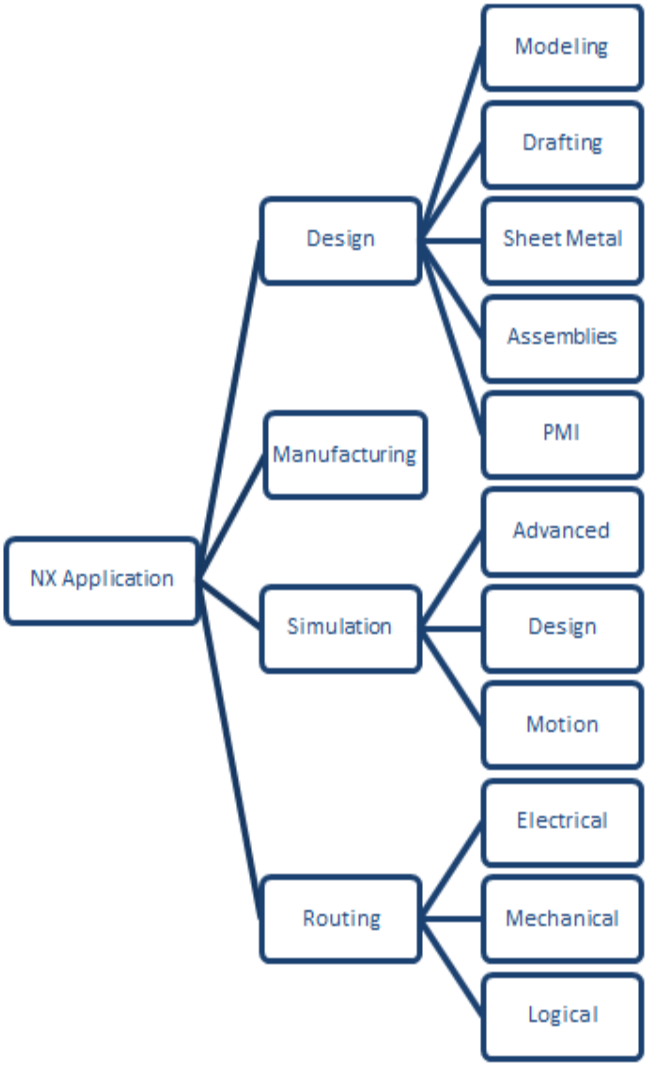


Figura 1 NX Applications. (Fuente: Elaboración propia).

En el entorno de diseño, se puede reproducir en el módulo de modelado 3D (*Modeling*) cualquier componente necesario para el desarrollo del producto. Tal y como se muestra en la Imagen 2. Así mismo, con la herramienta de dibujo en plano (*Drafting*) se puede realizar un plano 2D asociativo del modelado anterior con todas las especificaciones de materiales, tolerancias y métodos de fabricación, como se puede ver en el ejemplo de plano de la Imagen 3

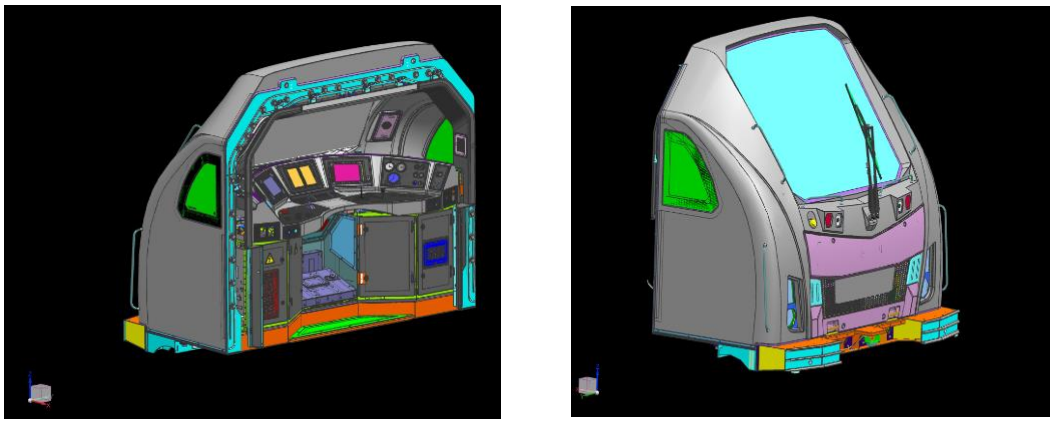


Imagen 2 Diseño 3D Cabina y Bastidor Locomotora. (Stadler Rail Valencia S.A.U., 2017)

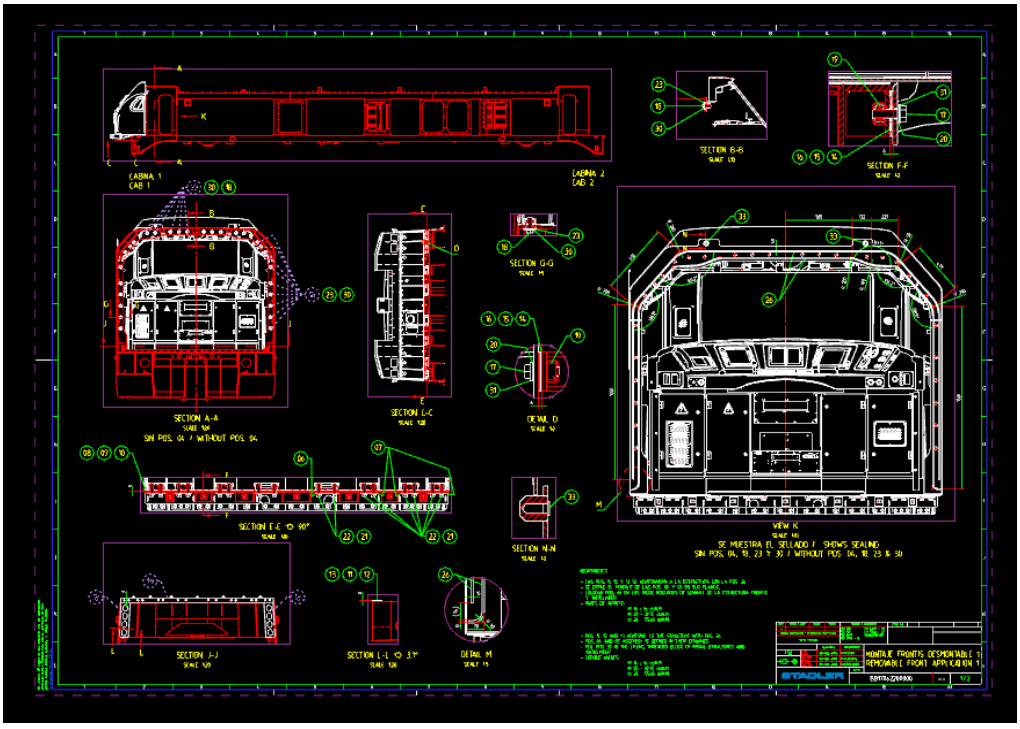


Imagen 3 Plano Cabina y Bastidor de Locomotora. (Stadler Rail Valencia S.A.U., 2017)

La herramienta de Chapa (*Sheet Metal*), permite diseñar las chapas metálicas necesarias y el modo de pliegue de las mismas para formar otros componentes. La unión de los modelos generados individualmente para formar componentes más complejos se realiza en la herramienta de Ensambles (*Assemblies*), donde se puede llegar a formar mecanismos como el de la Imagen 4. Por último, dentro de este entorno, destacamos la herramienta *Product & Manufacturing Information* (PMI) la cual posibilita realizar acotaciones sobre un elemento tridimensional en vez de únicamente poder detallar cotas u otra información relevante para la fabricación en los planos bidimensionales. (SIEMENS PLM, 2017)

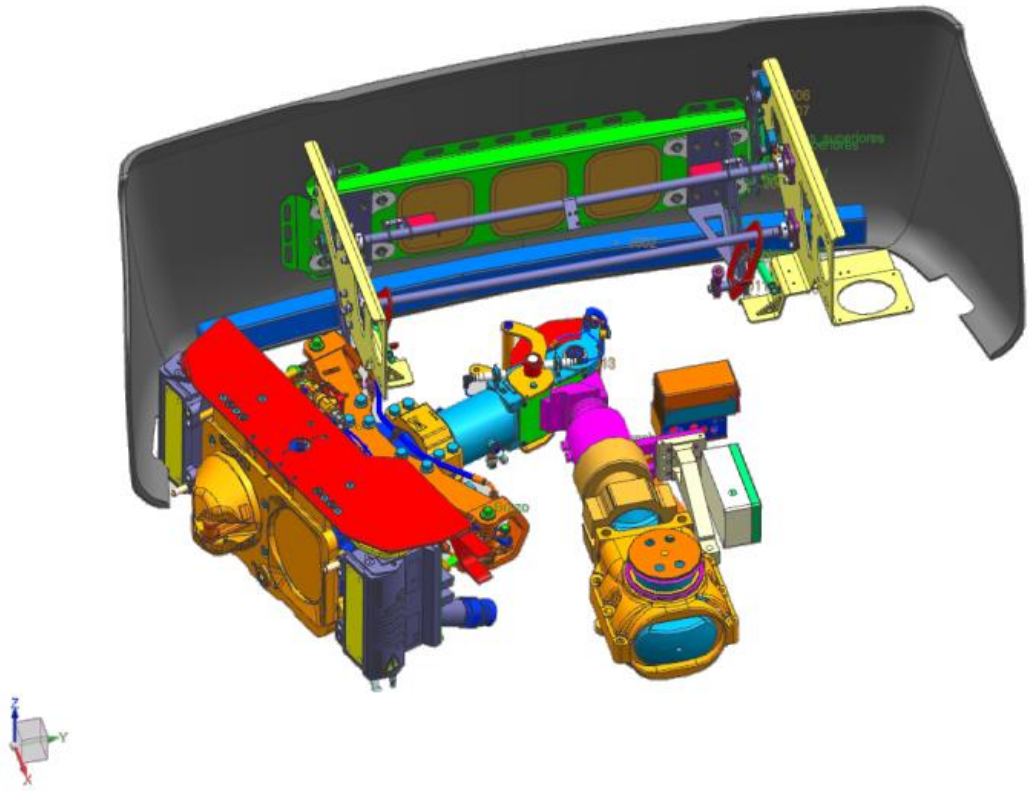


Imagen 4 Mecanismo de enganche de vagones. Parte frontal. (Stadler Rail Valencia S.A.U., 2017)

El entorno de Fabricación (*Manufacturing*) permite programar y postprocesar interactivamente los distintos métodos de mecanizado de las piezas como el fresado, torneado, taladrado, etc. De esta manera se establece la forma exacta de fabricación de una pieza con un rango de tolerancia mucho menor y por ello un margen de error menor, lo que hace que la producción sea mucho más precisa y haya menos problemas en el montaje y ensamblado del producto.

Dentro del entorno de Simulación encontramos herramientas como la simulación avanzada (*Advanced*) en la que se pueden emplear distintos *Solvers*. Un *Solver* es el método matemático de cálculo escogido para llevar a cabo el ensayo virtual. El *NX Nastran* es un *Finit Element Analysis* (FEA) que realiza cálculos en la estructura del producto mediante la división del volumen o la superficie a analizar en pequeños trozos. El cálculo se realiza de forma individual sobre cada trozo tantas veces como número de trozos se haya hecho. En la Imagen 5 se ve un ejemplo de estructura de cabina de una locomotora mallada para realizar el cálculo de estado de cargas límite de esta forma. Este método de cálculo se denomina cálculo por elementos finitos. Explicado de otra manera, es dividir un problema en muchos problemas más sencillos. Gracias a esta herramienta, se puede saber si la estructura del producto aguantará los estados de carga a los que estará sometido.

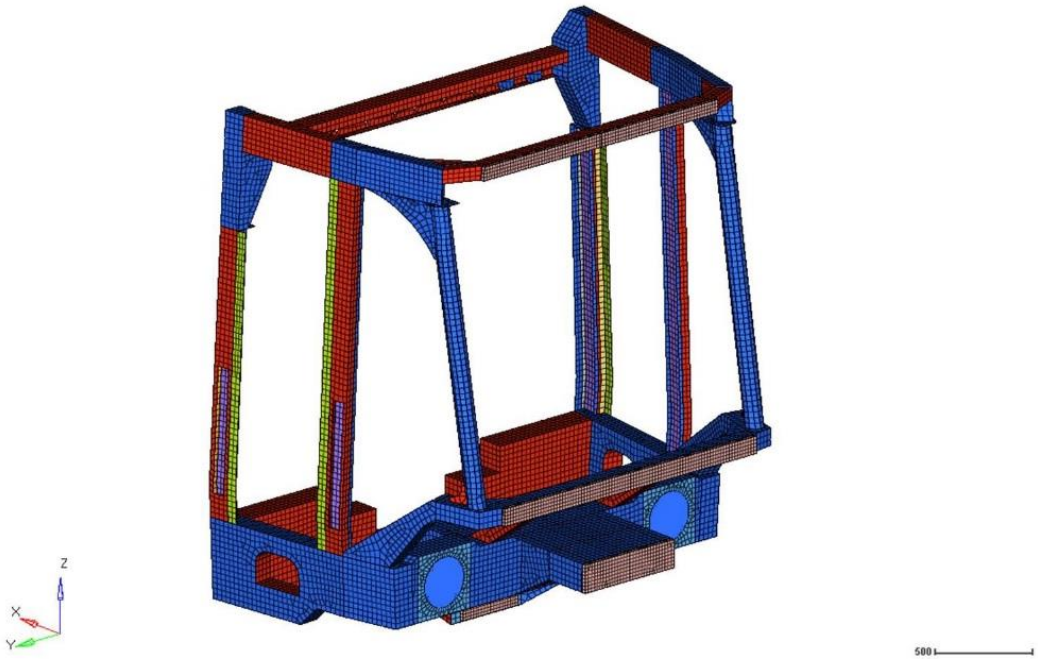


Imagen 5 Estructura cabina analizada con FEA. (Stadler Rail Valencia S.A.U., 2017)

Si no cumple normativa, se realiza un cambio en el diseño hasta que se cumpla. Para finalizar, se hace una comparativa entre los resultados de los ensayos virtuales y los resultados de los ensayos reales para observar las diferencias entre ellos y, de esta manera, ajustar el ensayo virtual según esas diferencias para que sea lo más parecido posible a la realidad.

Otra herramienta es la denominada *NX Thermal/Flow*. Esta herramienta es el *Solver* encargado de realizar los cálculos de temperaturas y flujos de forma individual o conjunta. Al igual que el *NX Nastran*, el *NX Thermal/FLow* es un FEA, pero en este caso, no se analiza las cargas en la estructura, si no, todo lo referente a flujos o temperaturas. Este *Solver* calcula, para cada trozo, la velocidad, presión, volatilidad y pérdidas de presión entre otras variables, de un flujo específico. NX dispone de una biblioteca de flujos para los que se puede realizar el análisis. Se define la trayectoria del flujo en el componente a analizar y los distintos gradientes de temperatura y *NX Thermal/Flow* hace el resto. Como resultado del análisis crea los gráficos que el usuario necesita, y en el mismo componente 3D, donde se han especificado las condiciones de contorno, el *Solver* marca, en función de la variable seleccionada, los distintos gradientes de diferentes colores para que se vea el resultado de forma gráfica (se puede observar un ejemplo en la Imagen 6).

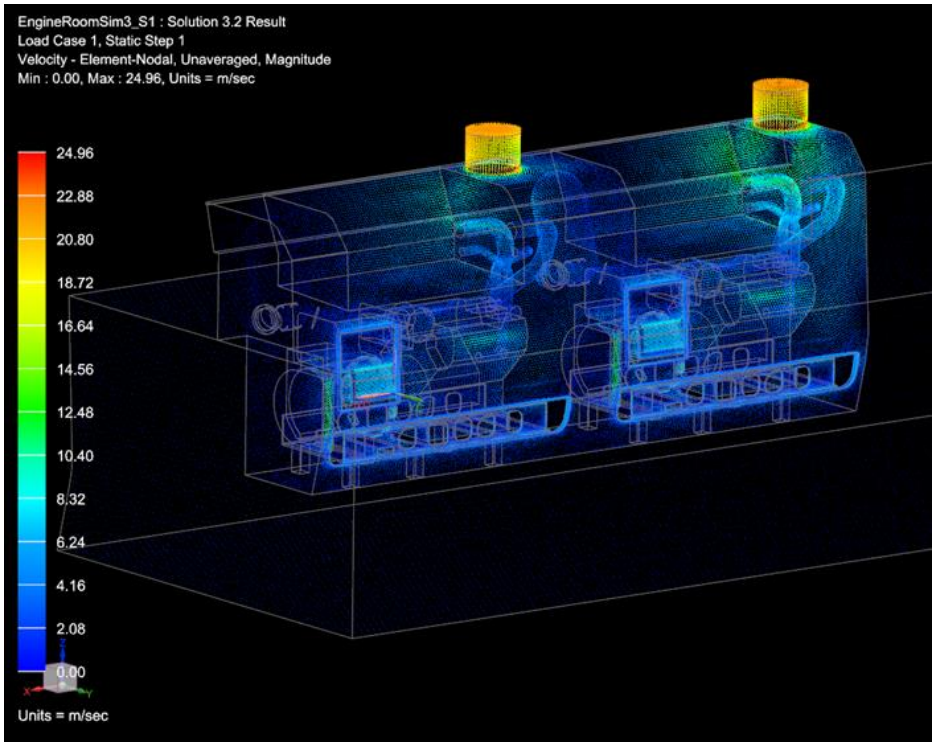


Imagen 6 Ejemplo de análisis *NX Thermal/Flow*. (Stadler Rail Valencia S.A.U., 2017)

También hay otro tipo de simuladores que realizan otro tipo de cálculos como son el de Diseño (*Design*) o el de Movimiento (*Motion*). Este último permite definir el movimiento que realizan los distintos componentes de un ensamblaje entre ellos o todos en conjunto y así ver de una forma gráfica si hay interferencias entre los componentes de un mismo ensamblaje, o entre ensamblajes distintos. De esta forma se puede averiguar el espacio exacto que necesita el componente para moverse libremente sin chocar con nada. Este espacio se denomina zona muerta. En la Imagen 7 se puede observar un ejemplo.

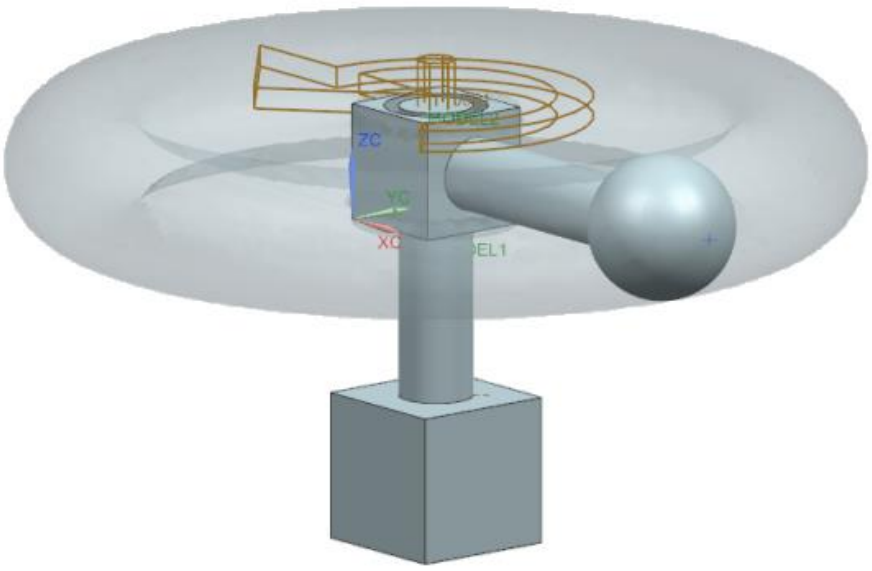


Imagen 7 Ejemplo de Zona Muerta. (Fuente: Elaboración propia).

Por último, el entorno de Enrutamiento (*Routing*) es el que facilita a los diseñadores establecer las líneas de corriente, el cableado interno, las tomas de luz o los conductos de agua para la refrigeración, tuberías de gas, conductos de combustible etc.

3.2.2. Modo de funcionamiento de *NX 10 Design*

Este apartado se va a centrar en explicar cómo trabaja el módulo de diseño de NX 10 ya que es el entorno en el que se ha integrado el trabajo realizado en el presente TFG, y que mejora la eficiencia de la herramienta *Update Structures*.

NX 10 Design permite genera tres tipos distintos de ficheros: El modelo 3D, el plano 2D y el ensamble de los modelos 3D para formar un mecanismo. De los dos primeros se han mostrado ejemplos en puntos anteriores. Se define como ensamble a la unión de piezas para formar un producto. Sin embargo, un ensamble también puede estar formado de la unión de otros ensambles. Por lo tanto para facilitar la comprensión, cuando se utilice **pieza**, se hará referencia a un único componente del conjunto. Cuando se utilice **conjunto**, se debe entender como unión de **piezas** que forman un mecanismo. Un ente es la unión de conjuntos y piezas que forman ya una parte importante del producto final, como sería por ejemplo el motor de la locomotora, la estructura, la mesa de mandos, etc. Por último, cuando se utilice **unidad de obra** significará conjunto de **entes** que forman el ensamblaje completo.

El módulo *design* trabaja mediante árboles de niveles para que, de una forma gráfica, se visualice la estructura completa del producto. Las distintas ramas del árbol equivalen a diferentes niveles de la estructura. A continuación, se pone un ejemplo de árbol estructural de un motor de coche. Se comienza con el diseño en NX de elementos sueltos como la biela, el pistón, el cigüeñal, etc. Tras su modelado individual, se procede a generar el archivo donde se guardará el subensamblaje de los diferentes elementos, obteniendo un conjunto que formará parte del ensamblaje final del motor. Por último, la unión de este ente con otros entes como el de las levas o la culata, forman el motor o ente del coche. En la Figura 2 se puede observar el árbol estructural del ejemplo del motor de coche.

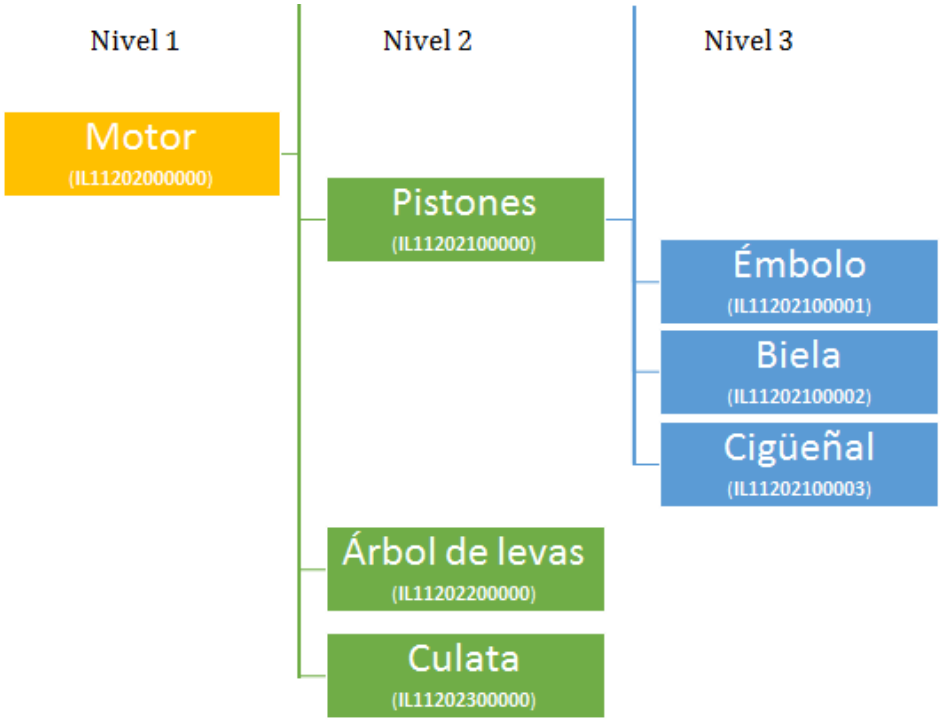


Figura 2 Árbol estructural motor de coche. (Fuente: Elaboración propia).

En este ejemplo se puede ver que el primer nivel se asigna a lo que sería la unidad de obra, y se continúa asignando niveles en orden descendente. De forma coloquial, también se emplea la siguiente nomenclatura: al elemento amarillo (motor) se denominaría el padre de los verdes, los pistones serían el padre de los azules siendo a su vez estos últimos los hijos de los verdes y éstos de los amarillos. En la empresa, el árbol estructural completo de la unidad de obra se llama **HOJA 0**. En base a ese documento gira la producción de las locomotoras, por lo que es fundamental mantener la estructura actualizada en tiempo real si fuera posible. Sin embargo, la tarea de comprobar los cambios y añadirlos es más complicada de lo que puede parecer, ya que una locomotora puede estar formada por alrededor de 8.000 componentes distintos. Por ello, el programa NX facilitó esta tarea mediante la herramienta *Update Structures*, que como su nombre indica, actualiza la estructura. En el siguiente capítulo se va a describir el funcionamiento de esta herramienta, porqué es necesario actualizar la estructura y cómo se lleva a cabo.

Así mismo, en la Figura 2 se ha asociado a cada pieza, conjunto o ente una codificación diferente que servirá para localizar de forma sencilla los distintos componentes en la base de datos. Esta codificación utilizada en *Stadler Rail* está compuesta por 13 dígitos. Los dos primeros son las iniciales del proyectista o diseñador que está realizando dicha pieza, aunque una vez finalizada, esas dos letras pasan a ser "BB". Los tres dígitos siguientes indican el número del proyecto, los tres siguientes el ente al que pertenece dicha pieza y el resto, el número de componente, tal como se puede ver a en el ejemplo que se muestra a continuación (Imagen 8).

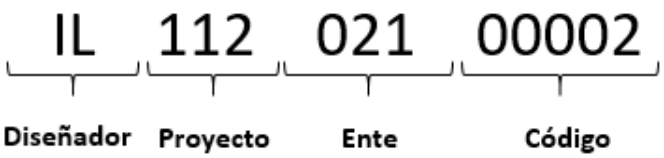


Imagen 8 Ejemplo codificación Stadler Rail. (Fuente: Elaboración propia).

La codificación de la unidad de obra tiene ceros tanto en los dígitos de ente como en los dígitos de código y se compone únicamente del número de proyecto (ejemplo del coche sería BB11200000000). Y dicha unidad de obra se compone de tres grandes grupos: definición (BB112DEF00000000), fabricación (BB112FAB00000000) y misceláneos (BB112MIS00000000).

A demás de los trece dígitos de codificación, se añaden dos más que sirven para indicar la revisión del elemento. Por lo tanto, cuando la pieza se modifica, estos dos dígitos van numerando las veces que se ha alterado el modelado original de la pieza. Cuando no se ha realizado de momento ninguna revisión, se le asigna un "- -", y a partir de la primera modificación se le pone A0, luego B0, C0, etc. También suele añadirse el nombre de la pieza/componente.

Por lo tanto, la pieza se denominaría de la siguiente forma: IL11202100002/- -/BIELA.

Capítulo 4. **ACTUALIZACIÓN DE ESTRUCTURAS**

4.1. ¿Por qué actualizar la estructura?

Como se ha comentado anteriormente, los entornos PLM y ERP comparten datos entre ellos. En la compañía, y concretamente con la aparición del ESA, el paso de datos de *TeamCenter* a SAP se comenzó a realizar de forma automática mediante archivos xml. Sin embargo, antes de esta mejora, el paso de datos se realizaba de forma manual, lo que siempre conlleva cierto riesgo de error humano. A día de hoy, los fallos del pasado se continúan arrastrando. Por ello, se ha implantado en la empresa una política de actualización de datos para la erradicación de la información errónea.

Es fundamental que los datos con los que se trabaja en gran parte de los departamentos de la compañía sean veraces y reales, por lo que deben estar actualizados. Si algún componente de la estructura está desactualizado y otro trabajador lo abre para modificarlo, se tendrán dos archivos de la misma pieza completamente diferentes. De ahí la relevancia de la actualización.

4.2. ¿Cómo actualizar la estructura? *Update Structure*

Los conjuntos se desactualizan cuando una pieza que los integra se modifica, ya que en el entorno de NX se abre únicamente la pieza a modificar, y hasta que el padre que contenga a ese hijo (pieza) no "vea" la modificación, mientras la otra está abierta, no se actualiza. Se puede ver en el ejemplo de la Figura 2. Si se modifica, por ejemplo, el diámetro de uno de los agujeros de la biela y la pieza se guarda y se cierra, tanto los pistones como el motor seguirán con el diámetro anterior, ya que el mero hecho de guardar no garantiza que se efectúe el cambio en todos los conjuntos en los que se utiliza una biela. Para actualizar la estructura completa primero se abre el modelado de la biela, luego el ensamble del conjunto de pistones y, por último, el ensamble del motor. De esta forma el nuevo diámetro se tendrá en cuenta en todo el conjunto del motor.

Ha de tenerse en cuenta que, si se abre el modelado de la biela y el ensamble del motor, pero no el ensamble de los pistones, el ente motor seguirá no estando actualizado, ya que la información de la actualización pasa a través de toda la rama del árbol estructural.

La solución al problema anterior que propone NX (el *Update Structures*) es la apertura de forma remota de todas las piezas, conjuntos y entes hasta completar la actualización de la unidad de obra. Una vez abierto todo, se debe hacer un guardado general y se cierra la herramienta. Como se ha comentado antes, una locomotora puede estar formada por alrededor de 8.000 piezas, por lo que el proceso es costoso, requiere mucho tiempo de computación y, al ser necesaria la apertura de las piezas en NX, se necesita una licencia de este programa, por lo que se inhabilita un puesto de trabajo. La actualización de la estructura no precisa supervisión a tiempo real y lo ideal sería llevarla a cabo al finalizar la jornada de trabajo para que al día siguiente todas las modificaciones se hayan analizado y actualizado. Por lo tanto, el ESA planteó un proyecto que consistía en generar una herramienta en la que se llevasen a cabo ciertas tareas que no requiriesen supervisión. El planteamiento del proyecto se basaba en realizar esos procesos a lo largo de la noche cuando ya no se anulara un puesto de trabajo para ello, y si era posible, realizando los mismos pasos que el *Update structure* pero más rápido y sin llegar a abrir las piezas del producto que no fueran necesarias.

Capítulo 5. **PROPUESTA DE MEJORA DEL SISTEMA (NIGHT)**

Se le puso el nombre de NIGHT al proyecto por el hecho de que se iba a ejecutar por las noches. Realizar la actualización sobre la marcha, al mismo tiempo que se decidía si era necesario abrir una pieza o no, era complicado, por lo que se tomó la iniciativa de realizar la programación en varios pasos. El primero de ellos consistía en generar un archivo, en base a la hoja 0, con la información necesaria para decidir qué piezas se abrían y no para llevar a cabo la actualización. A esta primera tarea se le llamó **Generación de Hojas 0**. El segundo paso, tras disponer de los datos generados en el primer paso, realizaba la apertura de los ficheros según se establecía también en el primer paso, y de esta forma realizar la actualización. Esta segunda tarea se llamó **Actualizar estructura**. En siguiente lugar, se desarrolló una aplicación que realizaba un análisis de intersecciones entre los distintos niveles de la estructura del producto, que son los componentes de la primera tarea, y que se denomina **Interferencias simples o compuestas**. Por último, aprovechando el archivo generado en la primera tarea, también se han implementado varias aplicaciones que facilitan la generación de los archivos de entrada de otras herramientas de la empresa. Cada aplicación se detalla a continuación.

5.1. Generación de Hojas 0

El proyecto del ESA comenzó con la investigación de una forma de actualizar la estructura sin la necesidad de abrir completamente toda la unidad de obra. Para ello se ha de explicar el tipo de componentes que forman el producto. A cada componente se le asignan unos Atributos que la definen y caracterizan. Algunos ejemplos serían: nombre del componente, código, tipo, propietario, fecha de última modificación, etc. La propuesta de la actualización se centra en el atributo Tipo. Dicho atributo clasifica al componente según la forma de abastecimiento, propio o externo, de un artículo y su utilización en la lista de piezas. En anteriores capítulos ya se ha hecho referencia superficialmente a esta nomenclatura. Los distintos tipos son:

- MPRI Materia Prima. Materiales elementales (chapas, perfiles, piezas en bruto, etc.) que siempre son de adquisición externa y han de ser manipulados para su conversión en un artículo de montaje. No deben tener componentes. Son componentes de los artículos semielaborados.
- EINC Elemento incorporable. Piezas elementales terminadas o conjuntos terminados que siempre son de adquisición externa y no han de ser manipulados (rodamientos, tornillos, escobillas, etc.). Son componentes de un artículo de nivel superior. No deben tener componentes.
- CONJ Conjunto. Cualquier elemento que puede ser fabricado o acopiado y que está formado por dos o más componentes (piezas y/o conjuntos). Son componentes de un artículo de nivel superior.
- ENTE Ente de diseño. Listas de piezas de diseño que forman parte de una Hoja 0.
- SEMI Semielaborado. En general, cualquier artículo a fabricar del que cuelga una y sólo una Materia Prima. Son componentes de un artículo de nivel superior.

- UOBR Unidad de obra y artículo del proyecto. La unidad de obra es el artículo principal de una obra y resultado final del proceso de fabricación. El artículo del proyecto es el artículo de primer nivel en la lista de piezas y agrupa los artículos correspondientes al conjunto locomotora (también UOBR), y los que se puedan crear para ensayos, repuestos, etc.
- DOCU Planos y documentos varios. Los planos y documentos, además de aperturarse como tales, también se dan de alta como artículos tipo DOCU. Realizando ciertas comprobaciones, se llegó a la conclusión de que para actualizar la estructura era necesario abrir los archivos que fueran ENTE y los archivos CONJ que no fueran de último nivel, es decir, que tuvieran hijos. Y se debía realizar de aguas abajo hacia arriba, es decir, empezando de los niveles más altos hacia los niveles más bajos.

En la decisión de las piezas que se debían abrir y las que no, en el primer planteamiento de la herramienta se abrían únicamente las piezas que eran ENTE. Sin embargo, al ejecutar el programa, la estructura no se actualizaba completamente. El error estaba en que las piezas modificadas que eran hijas de un CONJ no se tenían en cuenta, por lo que no se abrían y con ello los ENTES no modificaban su estructura añadiendo esos cambios. Por lo tanto, se llegó a la conclusión de que las que eran de Tipo ENTE debían abrirse en cualquier caso, y las que eran de Tipo CONJ sólo era necesario abrirlas si tenían hijos. Con esos datos, la aplicación recorría cada rama del árbol estructural hasta el final, recopilando la información que se muestra en el ejemplo de la Tabla 1.

Padre	Hijo	Nivel	Tipo	Nº hijos	Nº a cargar	Nº Entes
IL112FAB00000	IL11202000000	2	ENTE	3	3	2
IL11202000000	IL11202100000	3	ENTE	3	3	3
IL11202100000	IL11202100001	4	ENTE	0	0	0
IL11202100000	IL11202100002	4	ENTE	0	0	0
IL11202100000	IL11202100003	4	ENTE	0	0	0
IL11202000000	IL11202200000	3	...	...	...	...
IL11202000000	IL11202300000	3	...	...	...	...

Tabla 1 Datos a recopilar por el programa. (Fuente: Elaboración propia).

El funcionamiento detallado de la herramienta se expone en la Figura 6 mediante un diagrama de flujo junto con una explicación más técnica y detallada. Este va a ser el procedimiento para todas las aplicaciones desarrolladas en el presente TFG.

5.2. Actualizar Estructura

Esta segunda tarea es la que lleva a cabo la apertura de los componentes que, si se han modificado, desactualizan la estructura. De esta forma todos los conjuntos modificarán sus piezas para aplicar los cambios que se hayan llevado a cabo en alguno de sus componentes. Y así, también los entes, editarán sus conjuntos para aplicar los cambios que se hayan realizado en ellos.

El orden en que se abren los archivos también es importante. Deben abrirse desde el nivel mayor hacia el nivel menor, es decir, de forma ascendente refiriéndonos al árbol estructural, desde las piezas hacia los ENTES y de allí hacia la UOBR. De esta forma, tal y como se ha explicado anteriormente, la

modificación pasa por todos los elementos de la estructura. Se puede ver de forma gráfica en la Figura 3.

En el ejemplo del motor se abrirían las piezas, y luego los pistones. Si hubiese alguna modificación en las piezas la aplicaría a su archivo. Y por último se abriría el ENTE. Con la apertura de un ENTE, se realizará un guardado del mismo, lo que conlleva el guardado de sus respectivos componentes, en vez de guardar cada componente individualmente y luego el ENTE.

En el ANEXO IV se procederá a explicar en detalle los detalles técnicos de esta aplicación.

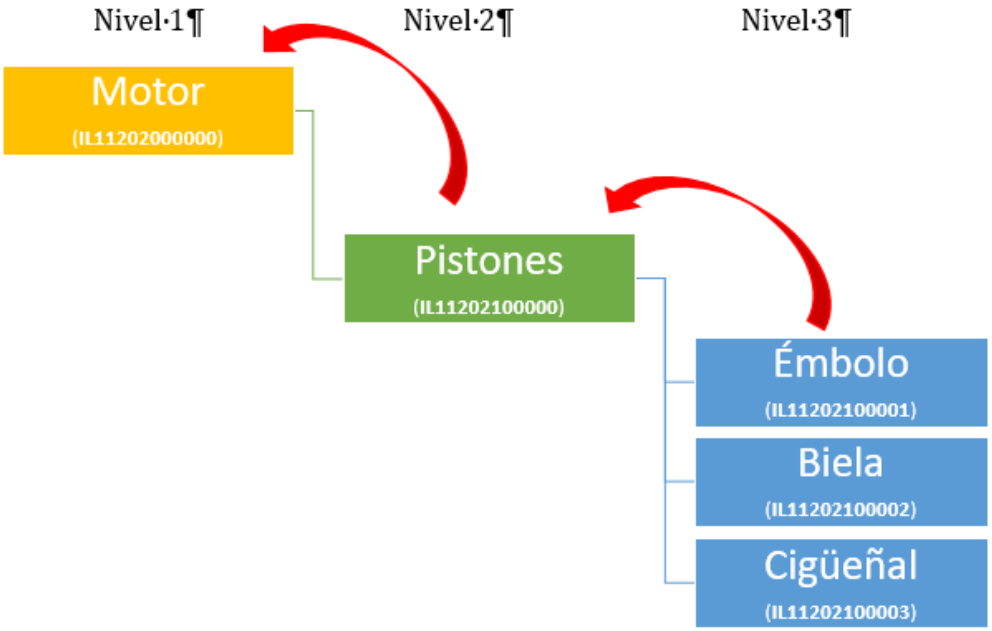


Figura 3 Trayectoria ascendente de la modificación (Fuente: Elaboración propia).

5.3. Otras aplicaciones relacionadas

También se realizaron ciertas mejoras a otras aplicaciones ya generadas en la empresa Para ello se utilizaron los archivos de Hojas 0 virtuales que resultaron ser muy útiles a la hora de realizar los cambios en dichas herramientas. En concreto, se mejoraron tres herramientas, como se ha dicho al inicio del presente TFG. La primera era una aplicación de NX que, especificando en el ensamblaje los componentes, realizaba un análisis de interferencias entre los mismos. La segunda era una aplicación de *TeamCenter* que, mediante la importación de un archivo de texto, asignaba permisos de edición de los archivos de diseño a las ingenierías externas a la empresa que se subcontratan para reducir la carga de trabajo de la ingeniería de la empresa. La tercera consiste en una herramienta implementada por otro trabajador de la compañía que realizaba una optimización de las piezas especificadas en un archivo de texto. En los siguientes apartados se explica su funcionamiento y de qué modo se han mejorado.

5.3.1. Interferencias simples o compuestas

El programa NX en el entorno de ensamblaje (*Assembly*) ofrece la posibilidad de realizar un análisis de intersecciones. Este análisis puede configurarse para realizar distintos estudios desde el mismo entorno de NX, seleccionando a mano los componentes que intervenían, indicando que se realice entre los componentes de sus subensamblajes (interferencia simple) o realizar el análisis de interferencias entre los subensamblajes que lo componen (interferencia compuesta). Es decir, en referencia a nuestro ejemplo del coche, se podrían hacer los siguientes análisis respecto al motor: el primer análisis sería, por ejemplo, biela con árbol de levas; el segundo análisis, las interferencias entre émbolo, biela y cigüeñal; y el tercero estudiaría las interferencias entre los pistones, el árbol de levas y el bastidor. También existe la posibilidad de realizar, en el mismo análisis, las interferencias entre los componentes del pistón, las interferencias entre los componentes del árbol de levas, las interferencias entre los componentes del bastidor y las interferencias entre pistón, levas y bastidor.

Por ello, y mediante esta aplicación de NX, se ha generado esta tercera herramienta que permite analizar los distintos tipos de interferencias entre componentes. Estas pueden ser: interferencia suave, interferencia fuerte o intersección completa. La primera significa que simplemente se tocan, superficie con superficie, como en la Imagen 9 abajo a la izquierda. La segunda significa que hay una intersección parcial, ejemplo en la Imagen 9 abajo a la derecha. La última significa que uno de los dos componentes se encuentra dentro del otro en su totalidad, es decir, interferencia total.

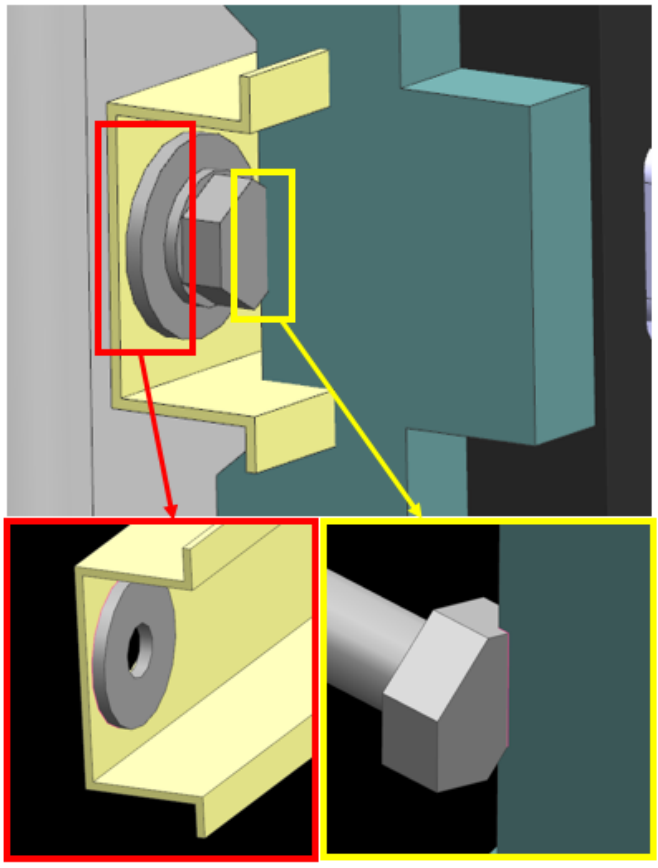


Imagen 9 Ejemplo Interferencia simple y fuerte parabrisas locomotora. (Stadler Rail Valencia S.A.U., 2017)

Como se observa en la Imagen 9, entre el componente amarillo y la arandela hay una interferencia superficial, lo que es normal, ya que debe existir contacto entre piezas. Sin embargo, entre la cabeza del tornillo y el componente verde hay una intersección entre ambos cuerpos que debe corregirse, ya que el tornillo no debe ocupar el espacio que ocupa el componente verde. Por ello, se debe localizar este tipo de taras en los modelados para corregirlos y evitar fallos en los mecanizados o montajes.

La aplicación que se introduce en este punto trata de realizar un análisis e informar al usuario de las interferencias fuertes entre componentes, las cuales deben modificarse en los conjuntos que se analizan para una mejor interpretación de los ensambles.

Dado que el análisis en sí ya lo realiza la herramienta NX 10, esta aplicación realizará las siguientes tareas: especificar qué tipo de análisis se quiere realizar para los componentes involucrados en dicho análisis, ejecutar la herramienta de NX que realiza los cálculos, volcar los resultados del/de los análisis en un fichero de texto y generar un informe que indique al usuario qué interferencias existen y de qué tipo, todo ello de forma remota y automatizada. También se da la posibilidad de realizar secuencialmente varios análisis especificándolos en el correspondiente fichero de entrada.

Se ha llegado a la conclusión de que, para analizar un proyecto entero, realizando el análisis de intersecciones simples en los ENTES de último nivel, es decir, en los ENTES que no tengan hijos ENTES, y el análisis de intersecciones compuestas en los ENTES no de último nivel, se debe realizar un análisis completo de la estructura del árbol. De todas formas, la aplicación que se ha implementado da la posibilidad de especificar un análisis únicamente de los componentes que se deseen en vez de realizar un análisis general basado en la Hoja 0 virtual del proyecto.

En el ANEXO V se procederá a explicar en detalle los aspectos técnicos.

5.3.2. Archivo de Permisos

La compañía *Stadler Rail* es una empresa internacional que, según la demanda del producto, puede llegar a tener una carga de trabajo excesiva. En dichas ocasiones, y cada vez con más frecuencia, la compañía subcontrata trabajos de diseño y cálculo a ingenierías externas. Actualmente, se pasa carga de trabajo a ingenierías externas como *Segula*, *Inhiset*, *Abegam* entre otras.

Para que usuarios externos a la empresa accedan a la plataforma del PLM, donde se dispone de todos los archivos de diseño, se ha de dar permiso de edición a dichos usuarios. Sin embargo, no se quiere dar permiso de edición en toda una UOBR, si no un permiso para uno o varios ENTES específicos a cada ingeniería subcontratada (subcontrata). Para ello, se ejecuta un programa donde, para cada ENTE, se detalla qué tipo de permiso se le da a cada subcontrata.

Existen tres tipos de permisos: el 2D (permiso de edición de planos), el 3D (permiso de edición de modelados) o el ALL (que da permiso de edición de todos los archivos de un componente). Para asignar los permisos, se debía modificar el fichero de la programación a mano. Dicho fichero es como el de la Imagen 10.

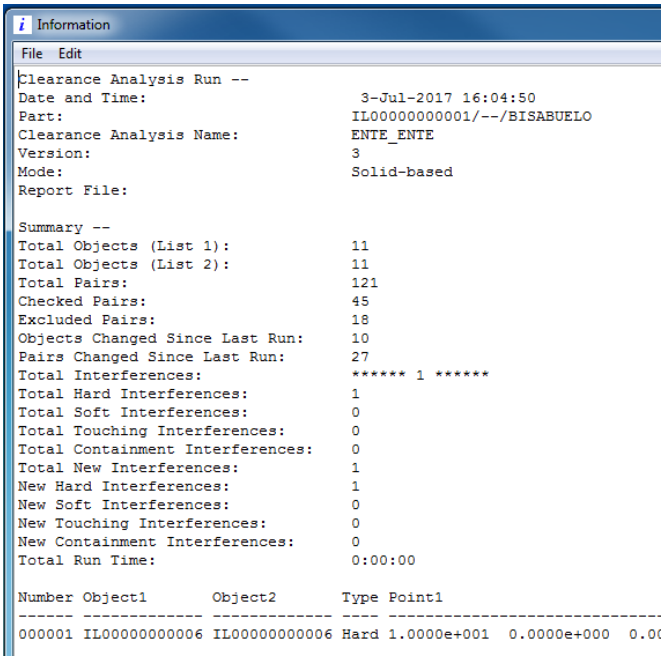


Imagen 10 Fichero de resultados de Interferencias. (Stadler Rail Valencia S.A.U., 2017)

Dado que la modificación del fichero manualmente era larga y tediosa, y siendo que dicho fichero asignaba permisos a lo largo de todo el proyecto se decidió realizar una aplicación que generase el fichero de la programación de asignación de permisos en base a la Hoja 0 de la UOBR deseada.

En el ANEXO VI se procederá a explicar en detalle los detalles técnicos.

5.3.3. Listado DQA

El *Digital Quality Application* (DQA) es una herramienta generada por otro trabajador de la empresa que utilizan todos aquellos usuarios que necesiten corregir de forma automática errores conocidos en los componentes, relacionados con las categorías de capas. También se deben corregir errores de menor gravedad como pueden ser el modelo de referencia incorrecto o la visualización de piezas ocultas.

Comienza con la comprobación de que realmente dicho componente existe, que existe su revisión y que ésta es correcta. También establece la capa 1 como la capa de trabajo y borra las capas antiguas o mal designadas con respecto a la normativa de capas de *Stadler Rail*. Finalmente se crea la distribución de capas oficial si se ha borrado la antigua. Dicha distribución oficial se refiere a qué tipo de componentes van en cada capa, tal como se muestra en la Tabla 2.

NOMBRE CATEGORIA	PRIMERA CAPA	ÚLTIMA CAPA
001.SOLIDOS	1	10
002.SOLIDOS-DETALLE	2	2
011.LAMINAS	11	20
021.CROQUIS	21	40
041.CURVAS	41	60
050.CDG	50	50
051.ENRUTAMIENTO-MECANICO	51	51
052.ENRUTAMIENTO-ELECTRICO	52	52
055.SERIGRAFIAS	55	55
061.DATUMS	61	80
101.PMI	101	110
255.NO-USAR	255	255

Tabla 2 Distribución de capas oficial (Stadler Rail Valencia S.A.U., 2017)

El DQA realiza más comprobaciones básicas y cambios para asegurar que todas las piezas sigan el mismo patrón en cuanto a distribución en capas, visualización por defecto (isométrica) o visibilizar los componentes que estén ocultos.

Para ejecutar el DQA, se necesita especificar qué componentes se desea que pasen por el mismo. Para ello se generaban manualmente los ficheros de texto de entrada. Ahora, la aplicación “Listado DQA” genera, según lo indique el usuario, un listado de los ENTES de último nivel del fichero “Generar Hojas 0” del proyecto deseado, o el listado de todos los componentes de la estructura en orden ascendente desde los niveles mayores hacia la UOBR. De esta forma se procesan primero los componentes más simples, de modo que en la cola del listado se encuentran los ensambles más complejos.

Capítulo 6. **RESULTADOS Y CONCLUSIONES**

Cada aplicación desarrollada en este TFG está enfocada a la mejora de un proceso o herramienta en particular. Dichas mejoras fuera de su entorno no tienen sentido, ya que se basan en la tarea a mejorar y, así mismo, utilizan recursos específicos de dichas tareas. Por lo tanto, y teniendo en cuenta este detalle, se procede a enumerar los resultados y conclusiones para cada aplicación de forma individual.

La primera aplicación que hemos denominado “Generar Hojas 0”, es la base sobre la que descansan las mejoras del resto de herramientas implementadas, ya que sin la generación de las Hojas 0 de las UOBR no sería posible haber realizado ninguna de las demás tareas, ya que todas utilizan los ficheros donde se vuelca la hoja cero virtual o árbol estructural.

La aplicación “Actualizar estructura” es el objetivo principal del presente proyecto, que consistía en encontrar una manera de realizar la actualización de los modelos utilizados en el departamento de ingeniería, pero de forma más rápida, sustituyendo así la aplicación nativa del PLM *Update structures*. Tras observar el funcionamiento de la aplicación implementada durante la realización de este TFG, en las pruebas llevadas a cabo a lo largo de dos meses, la mejora de la actualización de una estructura es aproximadamente de un 200%, ya que realiza la actualización en la mitad de tiempo. Sin embargo, para que los resultados de la eficiencia fueran concluyentes, se deberían realizar más pruebas a lo largo de un periodo de tiempo mayor. De todas formas, los resultados actuales son satisfactorios.

Los análisis de Interferencias simples y compuestos realizados de forma autónoma fue la última propuesta desarrollada del proyecto NIGHT y, por lo tanto, todavía está en fase de inicio de pruebas. Dicha aplicación se llevó a cabo a finales de mayo y principios de junio, y es ahora cuando ha comenzado a utilizarse. Los resultados, por el momento, también son satisfactorios, ya que anteriormente no se podía llevar a cabo los análisis de interferencias de forma automática. De esta forma, las interferencias detectadas hasta ahora por la herramienta generada en este TFG han sido corregidas por los diseñadores. Por lo que, en este aspecto, se va a reducir significativamente los errores por intersecciones de componentes.

Por último, la aplicación que mejora “Permisos” y “DQA“ genera, a partir de una serie de ficheros de entrada, una serie de ficheros particulares para cada una de estas dos herramientas, en vez de generar manualmente dichos archivos cada vez que se desee ejecutar una de estas aplicaciones.

En conclusión, el proyecto NIGHT desempeña satisfactoriamente todas las tareas que se han llevado a cabo en base a la primera aplicación desarrollada, “Generar Hoja 0” y la compañía *Stadler Rai*l puede realizar las tareas desarrolladas en el presente TFG en un horario fuera de la jornada laboral, ganando de esta forma la posibilidad de realizar las tareas automatizadas sin ocupar un puesto de trabajo y las licencias pertinentes.

ANEXO I. APLICACIÓN PRINCIPAL

En la Figura 4 se puede ver el diagrama de flujo completo de las aplicaciones desarrolladas durante este TFG. Cada aplicación se resalta en un color distinto para su mejor comprensión. La entrada a cada aplicación es común para todas (desde el menú de NIGHT). Sin embargo, cada aplicación sale de la herramienta por un proceso distinto. Esto se ha remarcado con las flechas de color rojo. En cada aplicación hay una, indicando por donde se sale del bucle correspondiente. Un ejemplo de estas salidas se puede ver en la Imagen 11.

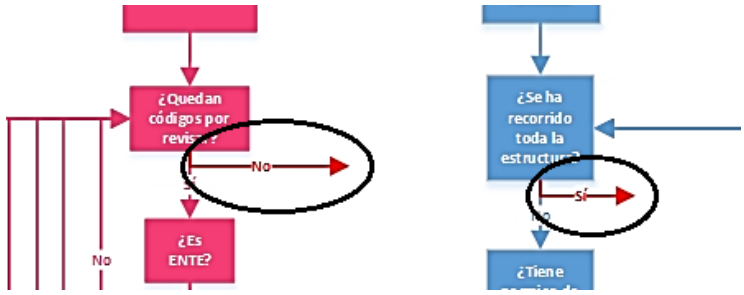


Imagen 11 Indicación de salida del bucle (Fuente: Elaboración propia).

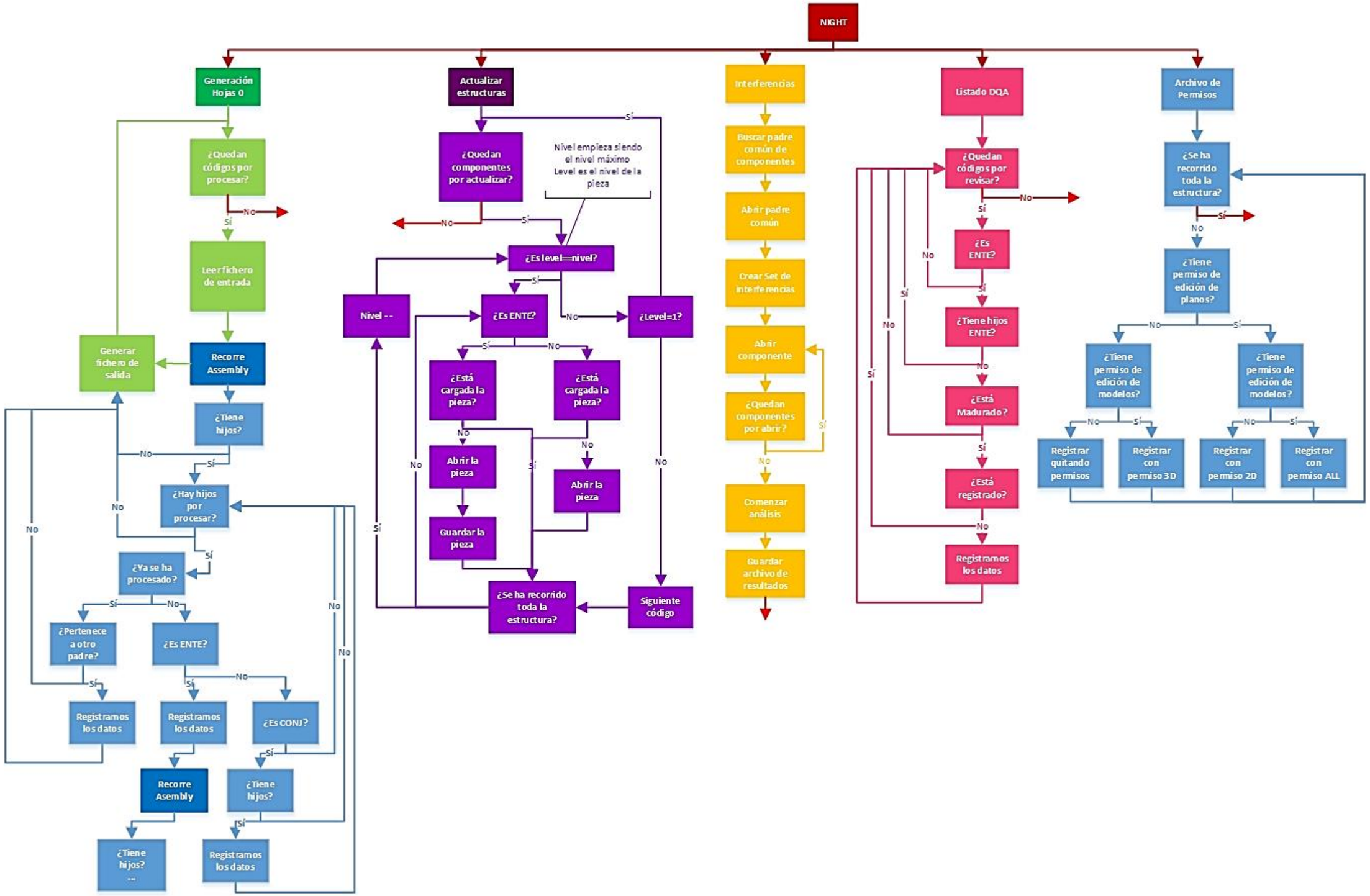


Figura 4 Diagrama de flujo completo de NIGHT (Fuente: Elaboración propia).

ANEXO II. FUNCIONAMIENTO DE LA APLICACIÓN PRINCIPAL

Una vez que el usuario se ha identificado, la aplicación NIGHT conduce hacia el menú principal, en el que se muestran todas las opciones que ofrece NIGHT. Estas opciones son las descritas en el Capítulo 5 como las aplicaciones que se han desarrollado en *Stadler Rail*. La pantalla de menú que se muestra al usuario se puede ver en la Imagen 12.

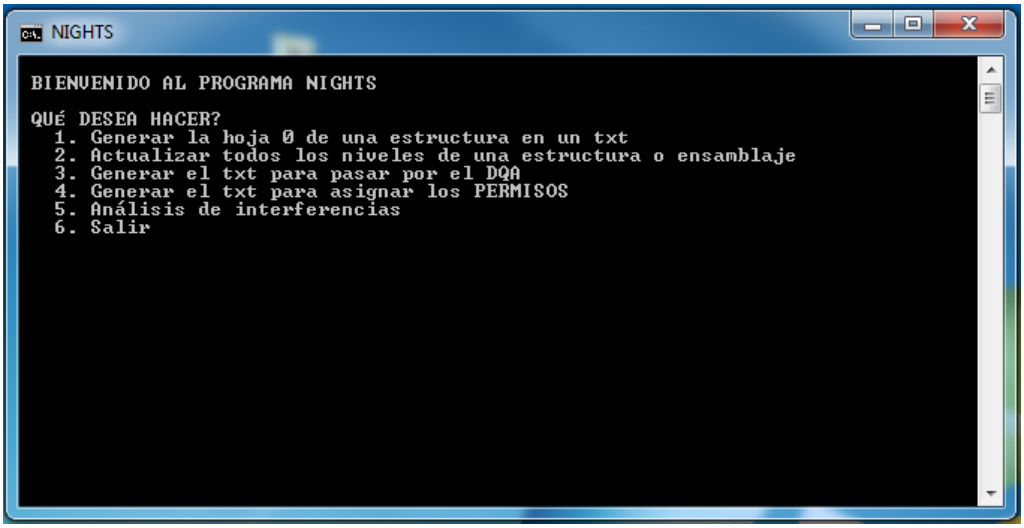


Imagen 12 Menú principal NIGHT (Fuente: Elaboración propia).

Tras realizar la selección de la operación que se quiere ejecutar, se muestra otra pantalla en la que se indican las especificaciones que la herramienta necesita para llevar a cabo la tarea. En cada aplicación se requiere de unos recursos distintos. A continuación, en la Imagen 13, Imagen 14, Imagen 15, Imagen 16 e Imagen 17, se muestra dicha pantalla para cada herramienta.

Para “Generar Hojas 0” se necesita un fichero de texto, ubicado en una ruta específica, que liste las UOBR o los componentes de los que se quiere generar la Hoja cero virtual o árbol estructural. Esto mismo se indica en la pantalla de especificaciones como en la Imagen 13.

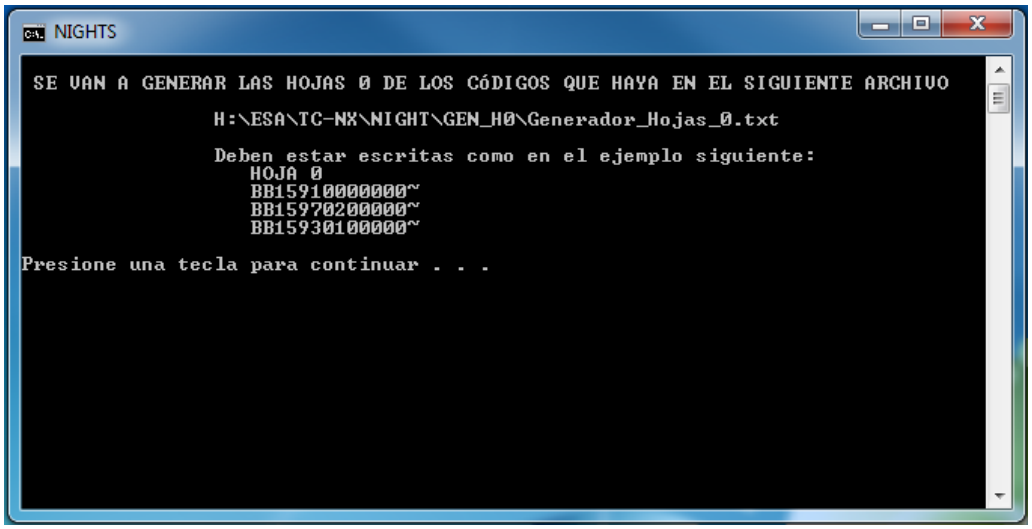


Imagen 13 Pantalla de especificaciones de "Generación Hojas 0" (Fuente: Elaboración propia).

En la Imagen 14 se detalla el formato en el que debe escribirse el fichero de entrada necesario para ejecutar “Actualizar estructuras”. En primer lugar, se especifica el código del componente que se desee actualizar su estructura y a continuación el código de la UOBR a la que pertenece.

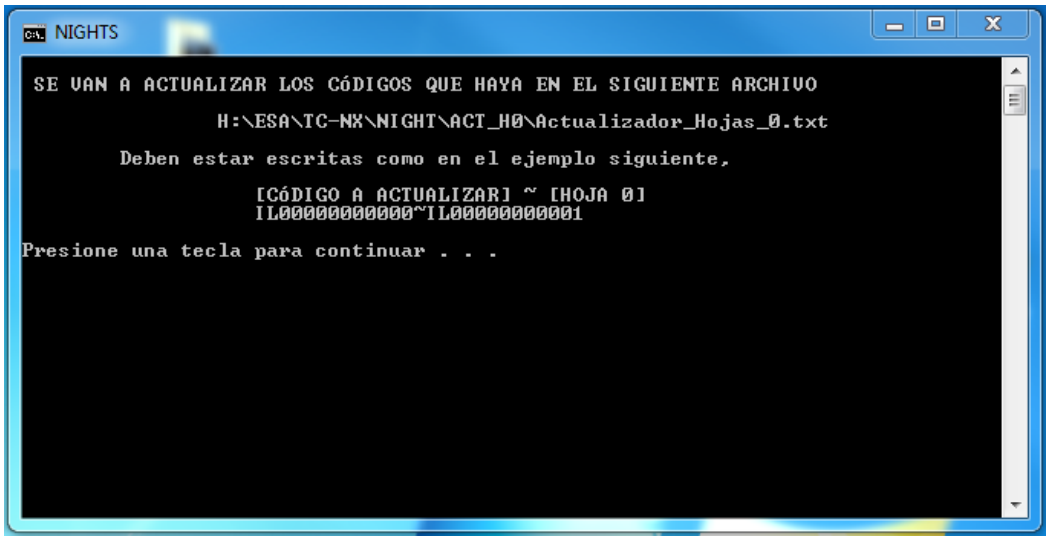


Imagen 14 Pantalla de especificaciones de "Actualizar Estructura" (Fuente: Elaboración propia).

La siguiente imagen (Imagen 15) pertenece a las especificaciones de “Interferencias”. Esta aplicación necesita un fichero de texto de entrada, siendo el nombre de este fichero el código de la UOBR donde se va a realizar el análisis. En ese fichero habrá un listado de los ENTES que vayan a participar en el estudio de intersecciones entre ellos. Sin embargo, si se desea realizar el estudio de las intersecciones entre los componentes de un ENTE, el fichero deberá contener únicamente el código de dicho ENTE.

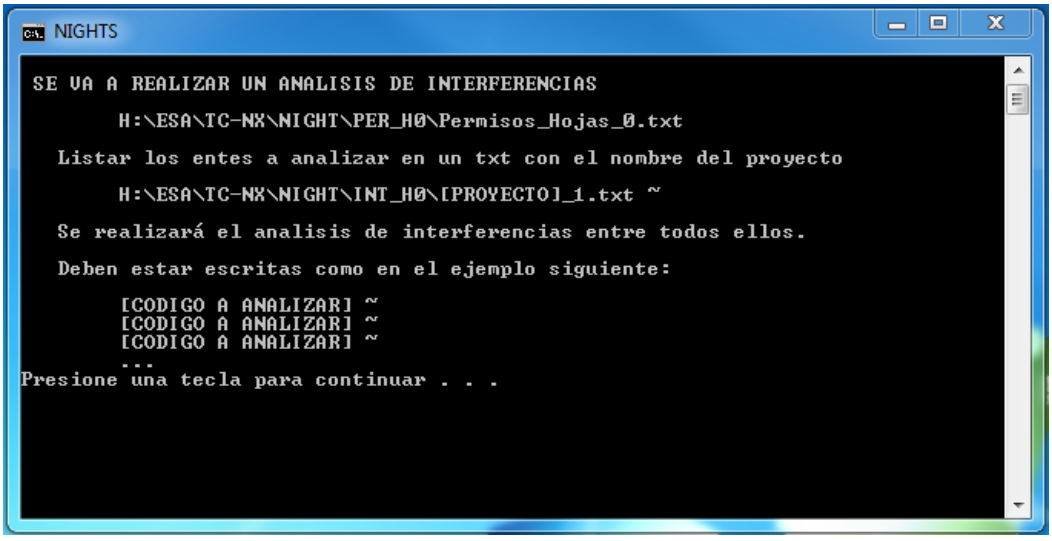


Imagen 15 Pantalla de especificaciones de "Interferencias" (Fuente: Elaboración propia).

Para generar el fichero y poder ejecutar la herramienta “DQA” se necesita un archivo de texto en la ruta especificada en el que se detalle el listado de los códigos a los que aplicarles el “DQA” y también, indicar la Hoja 0 virtual a la que pertenecen, como se muestra en la Imagen 16. Tras ejecutar la mejora

desarrollada, se obtiene un listado, en un archivo de texto, de los ENTES de último nivel del árbol que son los componentes a los que se les aplica el DQA.

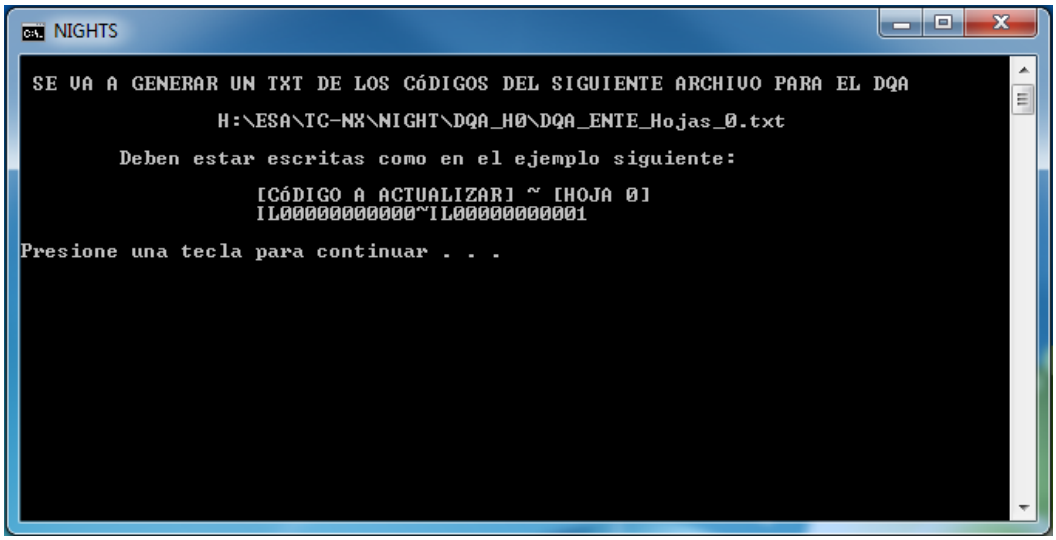


Imagen 16 Pantalla de especificaciones de "DQA" (Fuente: Elaboración propia).

En la aplicación de permisos se emplea un fichero de texto de entrada en el que se listan los nombres de las UOBR. Luego es necesario otro fichero dentro de la segunda ruta especificada, con el nombre del proyecto. En dicho fichero se debe especificar para cada ingeniería los ENTES a los que se les da acceso. En la Imagen 17 se observan estas especificaciones.

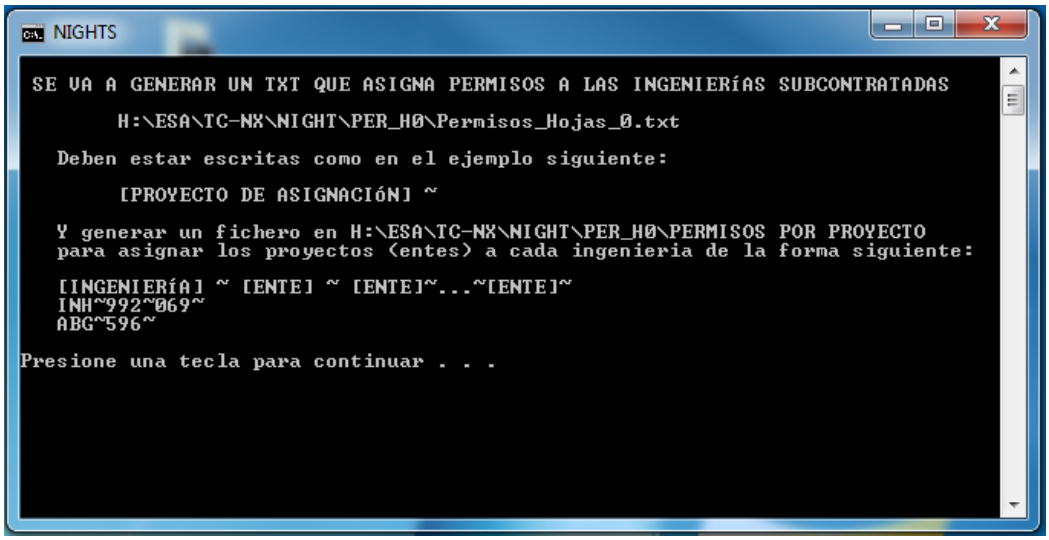


Imagen 17 Pantalla de especificaciones de "Permisos" (Fuente: Elaboración propia).

ANEXO III. DESCRIPCIÓN DE LA APLICACIÓN "GENERACIÓN HOJAS 0"

Como se ha comentado anteriormente, la tarea **Generación Hojas 0** es el primer paso o tarea de NIGHT. Esta primera aplicación comienza con el inicio de sesión en *TeamCenter* por parte del usuario que va a llevar a cabo la ejecución del programa. Este usuario no necesita tener ningún tipo de permiso de edición de archivos de NX ya que en esta tarea únicamente se recopila información y se vuelca en un fichero de texto. A continuación, tras la identificación del usuario, se procede a leer, de otro fichero de texto, las UOBR de las que el usuario quiera generar la hoja cero virtual o recopilación de datos para la actualización. Por ello, antes de ejecutar la aplicación, se debe especificar las UOBR en el mencionado fichero de texto. El programa generará la hoja cero virtual para cada proyecto especificado y lo hace de la siguiente manera (Figura 5):

Figura 5 Diagrama de flujo de Generación Hojas 0 (Fuente: Elaboración propia).

La lectura del fichero de entrada se realiza con una función básica de la biblioteca <stdio.h> llamada fopen.

La siguiente función que interviene es "*Recorre Assembly*", que se caracteriza por ser una función que se llama a sí misma, como se puede observar en la Figura 6. Este tipo de funciones se les llama funciones recursivas.

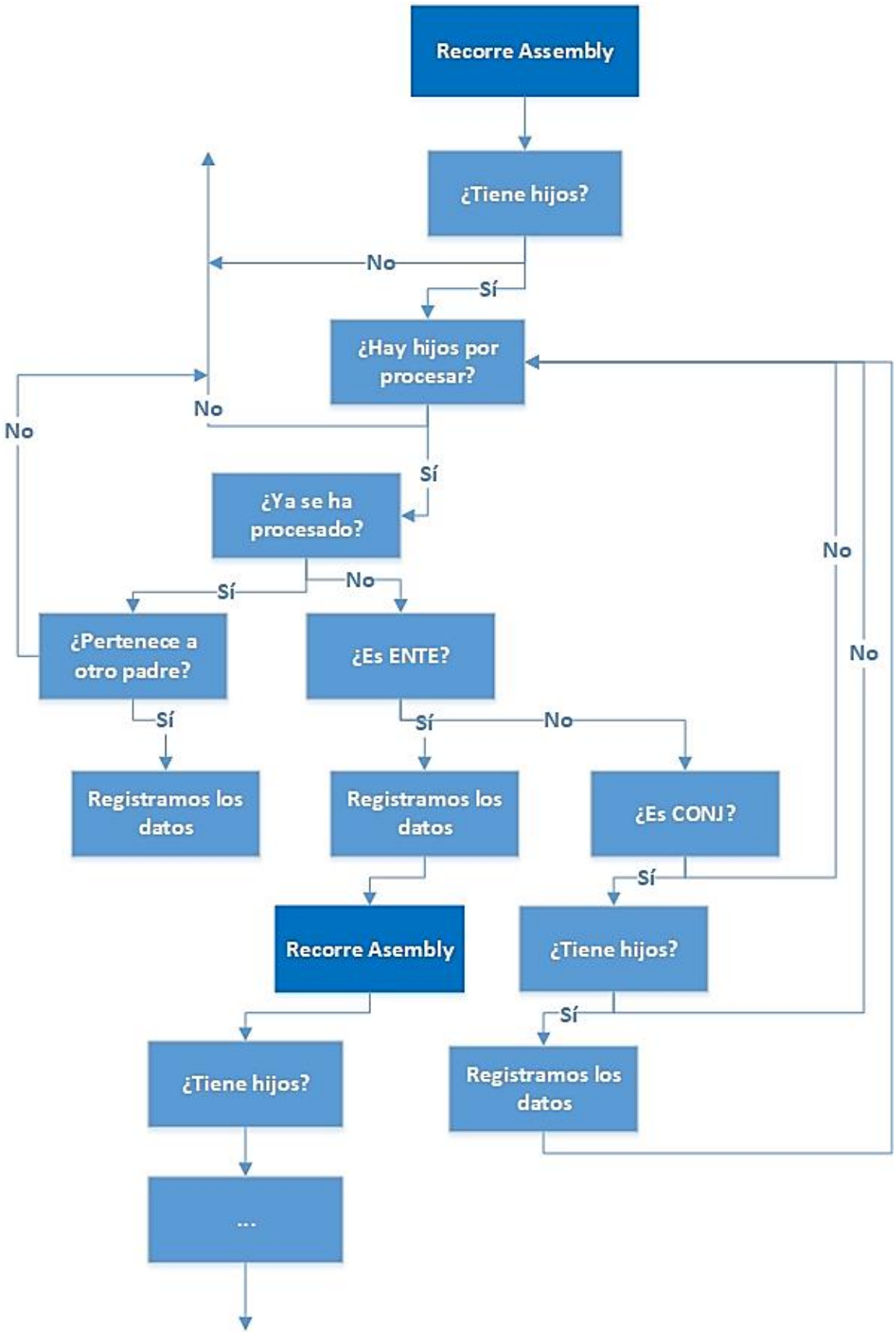
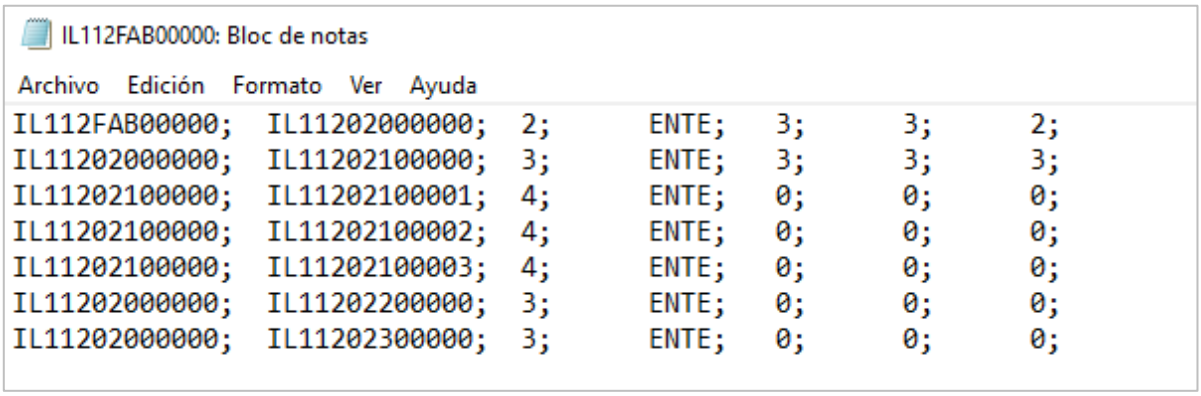


Figura 6 Diagrama de flujo de "*Recorre Assembly*" (Fuente: Elaboración propia).

Si lo que está analizando en ese momento la aplicación tiene hijos y son ENTE, se entra de nuevo en la función "*Recorre Assembly*", pero esta vez analizando a los hijos. Si estos hijos, también tienen hijos, entrará a analizarlos, y así sucesivamente. Cada vez que llama de nuevo a la función recursiva, se entra en un nivel superior de la estructura de la UOBR. Mientras, toda la información se guarda en una estructura de datos denominada struct, que es un tipo de variable que permite almacenar distintos datos de un mismo elemento. Es decir, guarda para cada ENTE el código del padre, su propio código, el nivel en el que se encuentra, el número de hijos que tiene, cuántos carga (analiza en el "*Recorre Assembly*") y cuántos son ENTES.

Tras recorrer toda la estructura y recabar toda la información necesaria, se vuelca la estructura de datos generada en un archivo de texto como el de la Imagen 18, separados siempre por un punto y coma para que se sepa cuándo acaba cada dato.


IL112FAB00000: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
IL112FAB00000;  IL11202000000;  2;      ENTE;   3;      3;      2;
IL11202000000;  IL11202100000;  3;      ENTE;   3;      3;      3;
IL11202100000;  IL11202100001;  4;      ENTE;   0;      0;      0;
IL11202100000;  IL11202100002;  4;      ENTE;   0;      0;      0;
IL11202100000;  IL11202100003;  4;      ENTE;   0;      0;      0;
IL11202000000;  IL11202200000;  3;      ENTE;   0;      0;      0;
IL11202000000;  IL11202300000;  3;      ENTE;   0;      0;      0;
```

Imagen 18 Ejemplo fichero de texto Hoja 0 virtual (Fuente: Elaboración propia).

ANEXO IV. DESCRIPCIÓN DE LA APLICACIÓN “ACTUALIZAR ESTRUCTURA”

La aplicación “Actualizar estructura” va abriendo los componentes especificados en la Hoja 0 virtual desde el nivel mayor hasta el menor. De esta forma primero se abren los ENTES de último nivel que ya no tienen hijos ENTES y de esta forma se actualizan los componentes que forman estos ENTES. Tras la apertura completa del último nivel del árbol, se va abriendo el resto de ENTES, por nivel, hacia la UOBR que es el conjunto de todos los entes y así las modificaciones suben a través de las ramas del árbol pasando por todos los niveles. Se diferencian dos tipos de apertura de fichero: los componentes que son ENTES se abren y se guardan para registrar todos los cambios, sin embargo, los componentes de tipo CONJ se abren, pero no se guardan ya que no es necesario. La apertura de los CONJ se realiza para que los ENTES de último nivel registren las modificaciones de todos sus componentes. Los CONJ, al realizar el guardado de los ENTES, también se guardan al formar parte de ellos.

Y esta operación se realiza tantas veces como proyectos se indican en el fichero de texto de entrada.

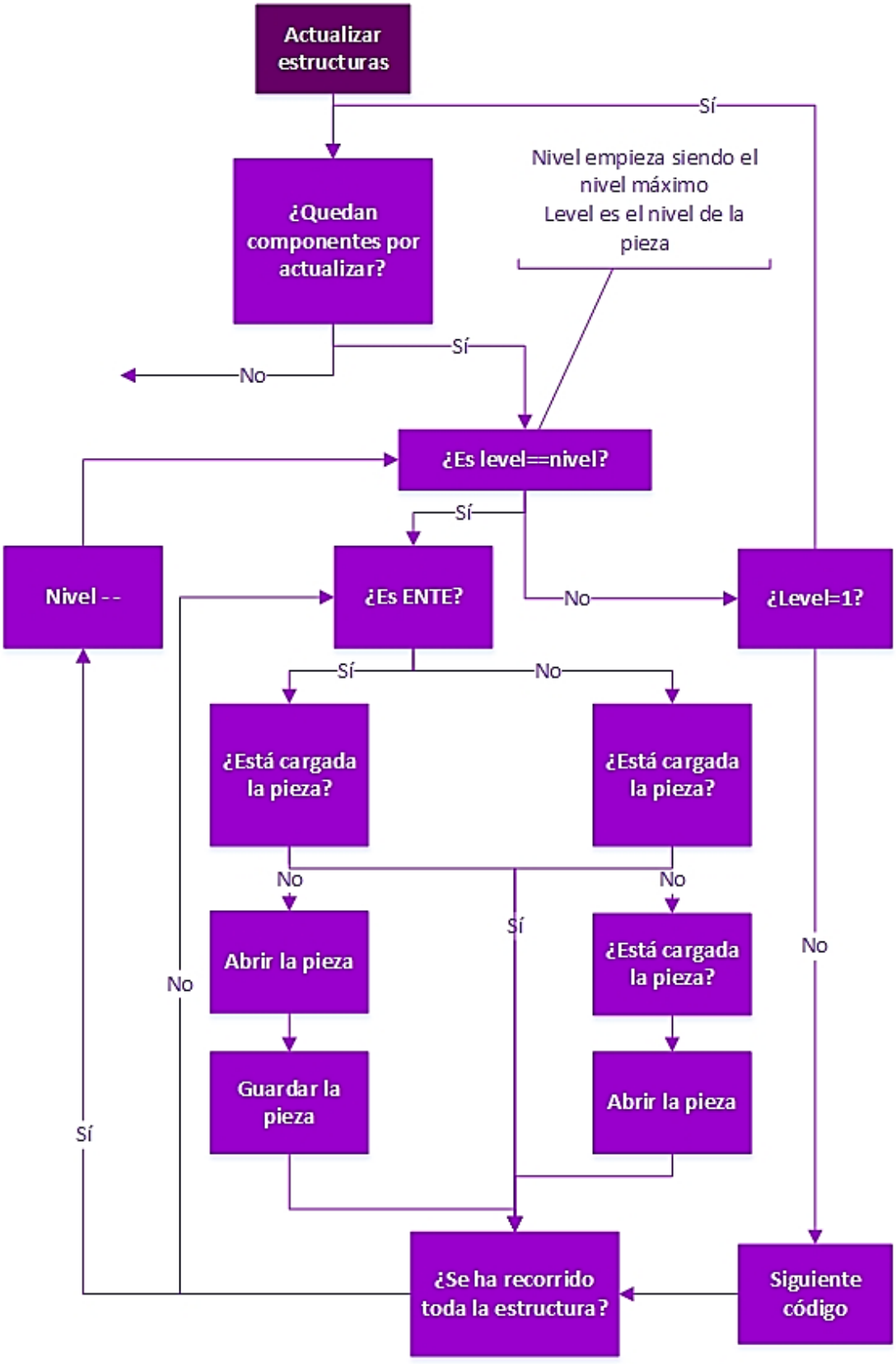


Figura 7 Diagrama de flujo de “Actualizar estructuras” (Fuente: Elaboración propia).

ANEXO V. DESCRIPCIÓN DE LA APLICACIÓN "INTERFERENCIAS SIMPLES O COMPUESTAS"

Al iniciar la aplicación, ésta lee del fichero de entrada los componentes que participan en el análisis, y tras localizarlos en la Hoja 0 virtual, se busca el primer componente que tiene a ambos en común (padre común). De esta forma, al abrir el ensamble y abrir únicamente ambas piezas, el análisis que se llevará a cabo sobre el padre, sólo buscará las interferencias de los hijos visualizados (abiertos).

El *Set* de interferencias se refiere a la configuración del análisis: si es simple o compuesto, si se incluye también el análisis entre las piezas del mismo componente, etc. Tras generar el *Set*, se lleva a cabo la apertura de los componentes deseados y se ejecuta el análisis. Para finalizar, se vuelcan los resultados en un fichero de texto para comprobar si hay interferencias fuertes que deban corregirse.

Figura 8 Diagrama de flujo de “Interferencias” (Fuente: Elaboración propia).

ANEXO VI. DESCRIPCIÓN DE LA APLICACIÓN "ARCHIVO DE PERMISOS"

En esta aplicación se recorre la Hoja 0 virtual buscando los ENTES especificados en el fichero de entrada, en los que se les debe dar permiso de edición a las subcontratas. Si el ENTE que se analiza está especificado en el archivo de entrada como editable por cierta subcontrata, en el fichero de volcado o de salida se especificará para ese ENTE un permiso 3D o ALL. Sin embargo, si el ENTE de análisis no está especificado en el archivo de entrada, se le asigna un permiso 2D (que permite únicamente visualizar los modelos, o modificar planos). Hay ciertos ENTES que nunca se les debe dar permiso de edición a las subcontratas al ser componentes patentados, por lo que en estos ENTES se les quita todo tipo de permisos. Un ejemplo de esto serían los ENTES *bogie*, que es la parte del vehículo ferroviario que permite el movimiento sobre las vías de la estructura y cabina. Al ser una parte del producto patentada, son pocos usuarios los que disponen de permiso para editarla.

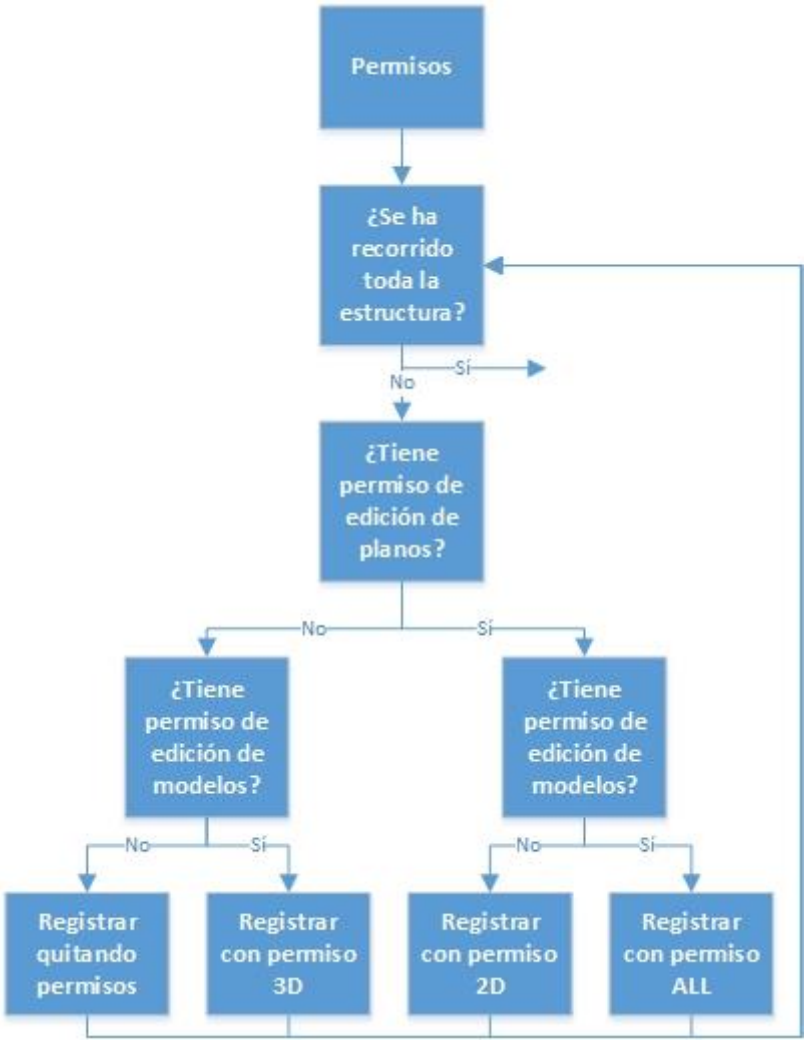


Figura 9 Diagrama de flujo de "Archivo de Permisos" (Fuente: Elaboración propia).

ANEXO VII. DESCRIPCIÓN DE LA APLICACIÓN “DQA”

Esta herramienta genera, en base a la Hoja 0 virtual, un listado de los componentes que cumplen ser de Tipo ENTE y que no tienen hijos, lo que los hace ENTES de último nivel, y estar madurados. Esto último significa que el componente ya no va a modificarse porque se ha finalizado su diseño. Los componentes que cumplen las tres condiciones se listan en un fichero de salida que después será el recurso de entrada para la herramienta “DQA”. En ese listado, cada componente aparece una vez, para evitar realizar las operaciones del “DQA” dos veces sobre el mismo componente.

Figura 10 Diagrama de flujo de “Listado DQA” (Fuente: Elaboración propia).

Capítulo 7. **BIBLIOGRAFÍA**

Chen, P., Zhu, W., & Ye, Z. (2012). Research and implementation of CAD-CAPP-PDM integrated system based on team center. *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, 320-323.

Chivaro Engineering Innovation. (Febrero de 2017). Obtenido de http://www.chivarotech.com/plm.htm

Experto PLM Siemens. (Febrero de 2017). Obtenido de http://www.expertoplmsiemens.unican.es/que-es-teamcenter.php

PDM software or PLM software: what's the difference? (Febrero de 2017). Obtenido de http://www.buyplm.com/plm-software/product-data-management-pdm-software.aspx

SIEMENS PLM. (Febrero de 2017). Obtenido de https://www.plm.automation.siemens.com

Stadler Rail Valencia S.A.U. (2017).

Sun, J., & Wang, Y. (2012). Research and Application of PDM and CAD Integration Technology. *2nd International Conference on Electronic & Mechanical Engineering and Information Technology (EMEIT-2012)*, 1562-1565.

PRESUPUESTO

PRESUPUESTO 05/07/2017

Nº de Actividad	Código	Descripción de las Unidades de Obra	Medida	Rendimiento	Precio unitario	Importe
01	01	**Trabajo llevado a cabo en la empresa**				
01.01	01.01	Implementación de las distintas aplicaciones				
	IIR	Ingeniero Industrial Responsable	100 h	1,00	46,44	4.644,00
	GITI-P	Ingeniero Industrial en prácticas	335 h	1,00	4,00	1.340,00
	%	Costes Directos Complementarios	1	0,01	50,44	0,50
		Coste Total				5.984,50
01.01	01.02	Reunión semanal con el Ingeniero Industrial Responsable para la decisión del formato de las aplicaciones a implementar				
	IIR	Ingeniero Industrial Responsable	15 h	1,00	17,50	262,50
	GITI-P	Ingeniero Industrial en prácticas	15 h	1,00	4,00	60,00
	%	Costes Directos Complementarios	1	0,01	21,50	0,22
		Coste Total				322,72
01.01	01.03	Reunión con el Ingeniero Industrial Responsable a causa de errores de compilación				
	IIR	Ingeniero Industrial Responsable	10 h	1,00	17,50	175,00
	GITI-P	Ingeniero Industrial en prácticas	10 h	1,00	4,00	40,00
	%	Costes Directos Complementarios	1	0,01	21,50	0,22
		Coste Total				215,22
01.01	01.04	Amortización de las licencias de los distintos programas implicados en la tarea				
	NXD	Licencia NX 10 Design	350 h	1,00	6,91	2.419,35
	NXP	Licencia Programación sobre NX	350 h	1,00	28,80	10.080,65
	%	Costes Directos Complementarios	1	0,01	35,71	0,36
		Coste Total				12.500,36
		Importe total				**19.022,79**

PRESUPUESTO	05/07/2017

Nº de Orden	Código	Descripción del Capítulo	Importe
01	01.	**Trabajo llevado a cabo en la empresa**	19.022,79

TOTALPRESUPUESTO EJECUCIÓN MATERIAL		**19.022,79**
Gastos indirectos	13%	2.472,96
TOTAL COSTE DE EJECUCIÓN MATERIAL		**21.495,75**
Beneficio industrial	6%	1.289,75
	Total	22.785,50
IVA	21%	4.784,95
TOTAL EJECUCIÓN POR CONTRATA		**27.570,45**

Asciende el presupuesto proyectado, a la expresada cantidad de:

VEINTISIETE MIL QUINIENTOS SETENTA EUROS CON CUARENTA Y CINCO CÉNTIMOS

04 de Julio de 2017