Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Aplicación MBSE al Sistema ATA 21: Modelado de Arquitectura en Capella y Simulación en MATLAB/Simulink

Autor: Miguel López Cordero

Tutor: Jesús Racero Moreno

Dpto. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2025

Trabajo Fin de Grado
Grado en Ingeniería Aeroespacial

Aplicación MBSE al Sistema ATA 21: Modelado de Arquitectura en Capella y Simulación en MATLAB/Simulink

Autor:
Miguel López Cordero

Tutor:
Jesús Racero Moreno

Dpto. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2025

Trabajo Fin de Grado: Aplicación MBSE al Sistema ATA 21: Modelado de Arquitectura en Capella y Simulación en MATLAB/Simulink

Autor: Miguel López Cordero
Tutor: Jesús Racero Moreno

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Inevitablemente, por mucho que hoy día se nos inculque bajo las sombras del individualismo y el liberalismo consignas del estilo «sigue tu camino» o «sé tú mismo», el calibre de las obras que uno realiza durante su vida es proporcional al grosor de las raíces que es capaz de echar en donde nace o crece.

Esas raíces son las que realmente definen la calidad y rigor que pueda llegar a tener este trabajo.

Estoy refiriéndome a mi familia, a mis amigos, a mi patria y a todas aquellas personas que han influido en mí y han hecho que este trabajo sea posible. Porque no basta con ser uno mismo, hay que estar dispuesto a echar raíces y por ello agradezco, por que no todos tienen la suerte de haber nacido en lugar donde yo lo hice.

Especialmente, quiero agradecer este trabajo, esta carrera a mi madre y a mi padre. Dos instituciones del trabajo duro, el esfuerzo y el más férreo estoicismo disueltos en la humildad y sencillez que siempre me quedarán marcadas.

Miguel López Cordero
Sevilla, 2025

Resumen

El presente Trabajo de Fin de Grado (TFG) se enmarca en la creciente relevancia de la **Ingeniería de Sistemas (IS)**, y en particular de la **Ingeniería de Sistemas Basada en Modelos (MBSE)**, como disciplina fundamental para abordar la complejidad inherente al desarrollo de sistemas tecnológicos avanzados en la industria actual. Se aplicará MBSE mediante la herramienta de **Capella** con la metodología **ARCADIA** en todas sus capas para obtener un modelo de arquitectura del Sistema de Presurización y Aire Acondicionado de Cabina (ATA 21) de un avión genérico inspirándose en el C-295.

Se recopilará la normativa de certificaciones de diseño de la EASA para el avión en el que se ha inspirado el autor, transformando esa normativa en **requisitos** que se introducirán al modelo. Se implementarán dos modelos de simulación programados en **MATLAB/Simulink** para la **Validación y Verificación** de requisitos definidos dentro del modelo de arquitectura. Uno hará un análisis **FTA (Fault Tree Analysis)** para sugerir la necesidad de redundancias y otro será un **modelo 1D** de un **ECS (Environmental Control System)** ya modelado anteriormente en Simulink para el **dimensionamiento** del sistema.

Se ha conseguido integrar todo, Arquitectura y modelos de simulación, mediante **Python4Capella**.

Abstract

This Final Degree Project (TFG) is framed in the growing relevance of **Systems Engineering (SE)**, and in particular of **Model-Based Systems Engineering (MBSE)**, as a fundamental discipline to address the inherent complexity of the development of advanced technological systems in the current industry. MBSE will be applied using the **Capella** tool with the **ARCADIA** methodology in all its layers to obtain an architecture model of the Cabin Pressurization and Air Conditioning System (ATA 21) of a generic aircraft inspired by the C-295.

The EASA design certification regulations for the aircraft on which the author has been inspired will be compiled, transforming these regulations into **requirements** that will be introduced into the model. Two simulation models programmed in **MATLAB/Simulink** will be implemented for the **Validation and Verification** of parameters defined within the architecture model. One will perform a **FTA (Fault Tree Analysis)** to suggest the need for redundancies and another will be a **1D model** of an **ECS (Environmental Control System)** previously modeled in Simulink for the **sizing** of the system.

The integration of all components—Architecture and simulation models—has been successfully achieved using **Python4Capella**.

Índice Abreviado

Índice

Índice de Figuras

1 Introducción

Me acuerdo de la primera vez que me presenté en el despacho de mi tutor de TFG. Tuvimos un breve coloquio algo animado, yo venía con la esperanza de empezar a trabajar en una dirección. Él mencionó muy entusiasmado 4 letras que me resonaron el resto de los meses: ”MBSE”. En un principio me pareció una cosa más dentro del universo extenso de conceptos que se dan en la carrera de ingeniería. Luego poco a poco me fue arrastrando una ola de interés y de curiosidad propia de la que sufre hoy todo el mundo con la IA. ¿La diferencia? Me encontraba en un camino mucho menos transitado. No obstante, ya sabemos que todas las tecnologías que vinieron a revolucionar el presente siempre fueron alguna vez veredas por las que de vez en cuando pasaba algún viajero con sus alforjas.

Por comenzar de alguna manera, se irán dando definiciones concisas de algunos de los conceptos que son necesarios para entender lo que es la Ingeniería de Sistemas y pronto todo esto irá relacionandose entre sí e se irá adentrándose en este singular campo. Lo primero que se menciona en el título de este trabajo es MBSE. Esto quiere decir *"Model Based Systems Engineering"*, o en español *"Ingeniería de Sistemas Basada en Modelos"*. Es una metodología que utiliza modelos para gestionar todo el ciclo de vida de un sistema (es decir un producto de ingeniería), desde la conceptualización hasta el desmantelamiento. Aunque pueda parecer un concepto abstracto, sus implicaciones son profundamente prácticas. A lo largo de este documento se desgranarán estos conceptos en detalle para revelar su verdadero potencial.

Para decirlo rápido, es una manera de simplificar sistemas complejísimos. Pongamos con un ejemplo, se puede pensar en un grifo. Un grifo está compuesto de las siguientes partes:

- **Mando(s):** Palanca, rueda o botón que el usuario manipula.
- **Válvula/Cartucho:** El mecanismo interno (cerámico, de compresión, etc.) que abre, cierra y regula el paso del agua (y mezcla en grifos monomando).
- **Cuerpo del Grifo:** La estructura principal que aloja la válvula y conecta las partes.
- **Caño/Boquilla:** Por donde sale el agua hacia el exterior.
- **Conexiones/Latiguillos:** Las partes que unen el grifo a las tomas de agua de la pared o encimera.

Esto es tan solo la arquitectura física del sistema, no se ha hablado todavía de las partes "interesadas" en el diseño del grifo, los stakeholders. El sistema "grifo" no existe de forma aislada; está inmerso en un contexto que incluye a todas las personas y entidades que interactúan con él a lo largo de su ciclo de vida. Esto abarca desde el diseñador que plasma la idea inicial considerando ergonomía y estética, los ingenieros que seleccionan materiales y mecanismos, el proveedor de la materia prima (como el metal), el fabricante que lo produce y ensambla, el instalador que lo conecta

a la red de tuberías, el usuario final que lo opera diariamente, hasta el personal de mantenimiento que podría repararlo o el gestor de residuos al final de su vida útil.

Tampoco se han considerado los requisitos que debe satisfacer. Funcionalmente, los requisitos clave son permitir el inicio y cese del flujo de agua bajo demanda, controlar el caudal de forma precisa, y en muchos casos, permitir la mezcla de agua fría y caliente para alcanzar una temperatura deseada, todo ello sin fugas cuando está cerrado. Pero igual de importantes son los requisitos no funcionales: debe ser duradero y resistente a la corrosión, seguro (materiales aptos para agua potable, sin bordes cortantes), fácil de usar e instalar, eficiente en el consumo de agua (cumpliendo normativas), estéticamente agradable y tener un coste de producción y venta competitivo.

Finalmente, el modelo dinámico del grifo se puede describir mediante casos de uso, funcionalidades detalladas, estados y eventos. Los casos de uso típicos serían "Lavarse las manos", "Llenar un vaso" o "Fregar platos", cada uno implicando una secuencia de funcionalidades como "Abrir flujo frío", "Ajustar caudal", "Mezclar con agua caliente", "Cerrar flujo". El ciclo de vida del grifo se puede modelar con una máquina de estados, pasando por estados como "Cerrado", "Abriendo", "Flujo Estable (Frío/Caliente/Mezclado)", "Cerrando". Las transiciones entre estos estados son provocadas por eventos, siendo el principal la interacción del usuario con el mando (girar, levantar, pulsar), pero también podrían considerarse eventos internos (desgaste de la válvula) o externos (cambio de presión de la red) en análisis más profundos.

Si se sigue una metodología (o reglas) a partir de un lenguaje para representar en modelos todas estas ideas que describen al grifo que se ha puesto sobre la mesa se obtendrá la tecnología de la que es estudio este TFG, el MBSE. Quizás todavía no se ha conseguido llamar la atención del lector lo suficiente, pero paciencia.

1.1 La Ingeniería de Sistemas Actual y su complejidad

Ahora bien, se está hablando tan solo de un simple grifo. Qué ocurre cuando se tiene un avión entero; ya no se tiene un sistema simple, sino 1000 sistemas distintos y que requieren un nivel de planteamiento, desarrollo y entendimiento infinitamente mayor. Dichos sistemas deben estar perfectamente integrados entre sí, y diseñados de tal manera que sus funciones individuales contribuyan de forma conjunta a la consecución de capacidades de nivel superior. Es aquí cuando la ingeniería de toda la vida, la que se basa en documentos queda muy anticuada y primitiva... El MBSE surge contra esta forma tradicional de hacer ingeniería de sistemas, para terminar englobándola.

Póngase que se está redactando un documento técnico que explica el diseño de un sistema de mi empresa ¿Qué ocurre si se quiere cambiar un requisito y este requisito aparece en 50 hojas distintas? ¿Qué ocurre si un requisito tiene un valor numérico y se quiere reducir ese valor de 20 kilos a 18, lo que se va a causar que ciertos componentes del sistema pasen a incumplirlo? ¿Qué ocurre si se añaden dos requisitos que dicen exactamente lo mismo, no se está siendo redundante? ¿No se debería avisar alguien de todos estos problemas?

¿No se deberían establecer reglas de validación inherentes al propio proceso de diseño de cualquier sistema?

Pues esto es un parte importante del MBSE, una forma de resolver muchos problemas que causan el trabajar con documentos. Ya no es solo que se trabaje con información de manera digital, eso se lleva haciendo durante 40 años... Si no que además se haga de manera integrada, trazable y validable en los distintos niveles que se da un problema.

La Ingeniería de Sistemas Basada en Modelos (MBSE) representa una evolución frente a los enfoques tradicionales que dependen de herramientas aisladas como hojas de cálculo y bases de datos de requisitos. El valor fundamental de MBSE reside en su capacidad para crear un modelo integrado donde la arquitectura, las funciones, los requisitos y los actores están interconectados como se muestra en la **Figura 1.1**. En ella se representan diferentes tipos de modelado (Diagrama de bloques, Modelado físico 1D, Geometría 3D) así como la integración de la arquitectura con

herramientas de simulación (como Simcenter Amesim, FEA, CFD) según las distintas capas de descomposición del sistema (Sistema, Subsistema, Componente). La simulación es una manera de V&V lo planteado en la arquitectura, como se ve en la imagen. Esta visión holística permite comprender la lógica funcional detrás de cada componente y sus relaciones, facilitando la validación y verificación en etapas tempranas del desarrollo. Al detectar inconsistencias o errores de diseño directamente en el modelo, se consiguen ahorros sustanciales, evitando que estos problemas escalen a fases posteriores donde su corrección sería mucho más compleja y costosa.

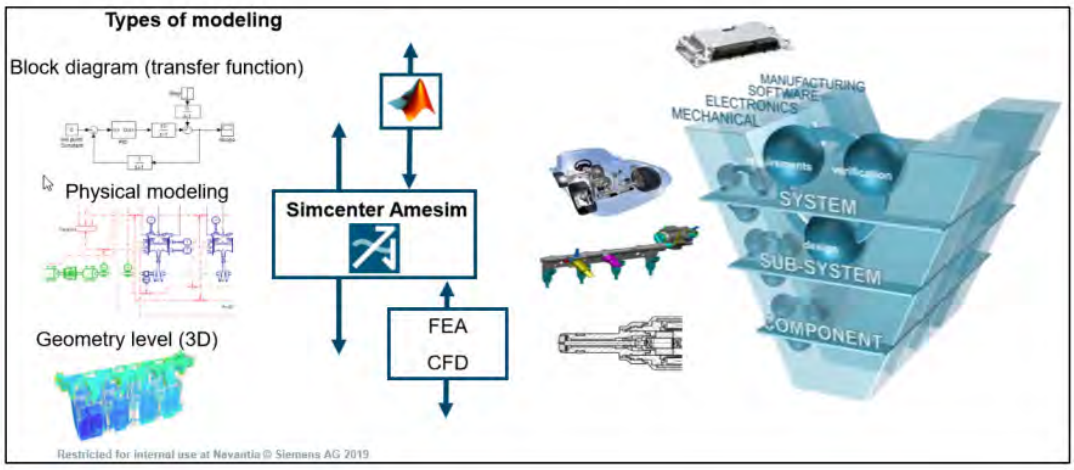

Figura 1.1 Ilustración de la integración entre modelado y simulación..

1.1.1 La Industria Aeroespacial, promotora del MBSE

Esta visión holística y conectada de los sistemas es lo que habilita uno de los mayores valores del MBSE: la posibilidad de realizar análisis, validaciones y verificaciones (V&V) de forma mucho más temprana y rigurosa dentro del ciclo de vida del desarrollo. Al poder evaluar la coherencia, detectar inconsistencias, simular comportamientos y verificar el cumplimiento de requisitos directamente sobre el modelo en las fases iniciales de diseño, se identifican problemas y errores que en un enfoque tradicional pasarían desapercibidos hasta etapas mucho más avanzadas.

La experiencia de empresas como Boeing con el desarrollo de sistemas aeroespaciales complejos, como el sistema de cableado y red digital del 787, ilustra este punto de forma contundente. La creciente complejidad de las redes embarcadas hizo inviable la gestión basada en documentos. La implementación del MBSE, mediante un modelo detallado que integraba arquitectura y señales, no solo permitió gestionar dicha complejidad, sino que, fundamentalmente, facilitó la detección temprana de errores de diseño.

Este es el núcleo del ahorro de costes que justifica la inversión en MBSE: corregir un error de diseño o una inconsistencia en el modelo durante las fases conceptuales o lógicas es órdenes de magnitud más económico que descubrirlo durante la integración de hardware y software, las pruebas de sistema o, en el peor de los casos, una vez el producto está en operación. Por lo tanto, MBSE se postula como una estrategia clave para desplazar el esfuerzo de detección y corrección de errores hacia las fases iniciales del ciclo de vida, reduciendo drásticamente los costosos retrabajos y modificaciones en etapas tardías, que son tradicionalmente las más pesadas.

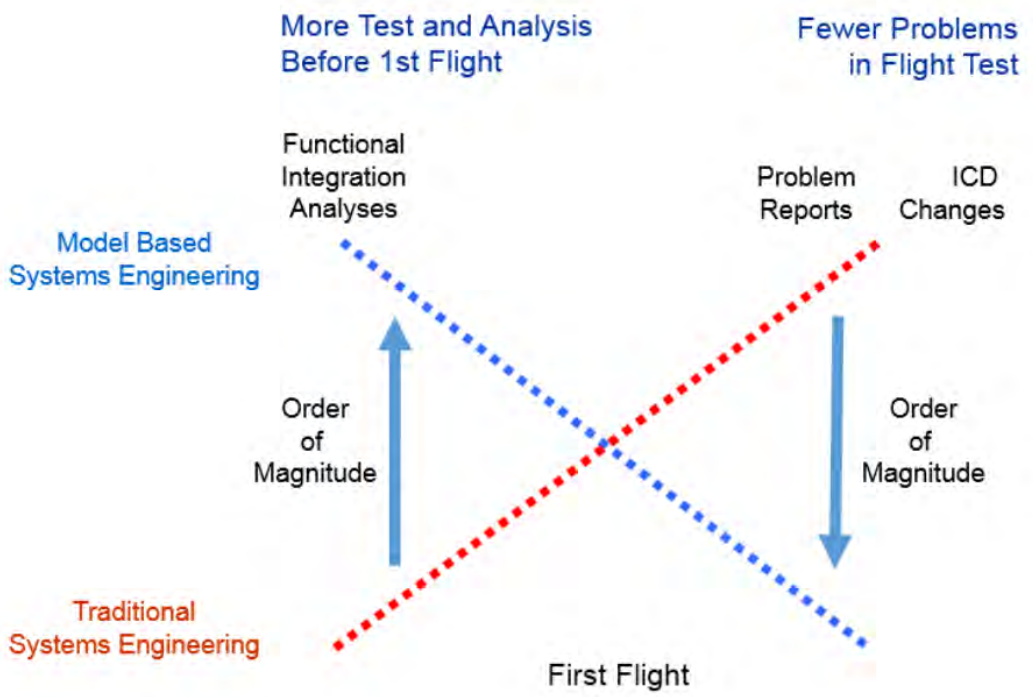

Figura 1.2 Representación gráfica comparativa entre MBSE y la ingeniería de sistemas tradicional en la detección de fallos funcionales[Massachusetts Institute of Technology, sf].

La evolución de las redes en los aviones Boeing (desde la A429 punto a punto, pasando por la AIMS compartida del 777, hasta la red Ethernet determinista ARINC 664 altamente compartida del 787) hizo que la gestión tradicional basada en documentos (como hojas de cálculo) fuera inviable. Especialmente con redes compartidas y nodos intermedios, manejar las interfaces y la propagación de cambios entre publicadores y suscriptores se volvió casi imposible debido a su complejidad.

Por ello, se construyó un modelo detallado que representaba todas las señales del sistema hasta el nivel de bit, integrando todos los componentes (LRUs) y la arquitectura funcional y lógica.

1.2 Solución MBSE

El MBSE se ha practicado durante más de 10 años en la industria aeroespacial, especialmente Thales Alenia Space (TAS) y Airbus Defence and Space (ADS), han sido quienes han desarrollado sus propias metodologías y cadenas de herramientas con mayor interés.

Thales desarrolló el Enfoque Integrado de Análisis y Diseño de Arquitectura, también conocido como ARCADIA, y Airbus creó la metodología Misión / Operación / Función / Lógica / Técnica, también conocida como MOFLT, y el Modelo de Arquitectura Central de Ingeniería de Sistemas (SECAM). Se ofrece una pequeña comparación de la estructura que sigue cada una en la **Figura 1.3**. La metodología ARCADIA también es utilizada por otras entidades en la industria espacial, ya que está implementada en la herramienta de código abierto Capella[1].

[1] El modelo de negocio de esta compañía entorno a Capella se centra en la venta de cursos, clases y certificaciones para el aprendizaje de la metodología ARCADIA, así como, la venta de Add-ons personalizados y específicos para el campo del que se ocupe el cliente interesado

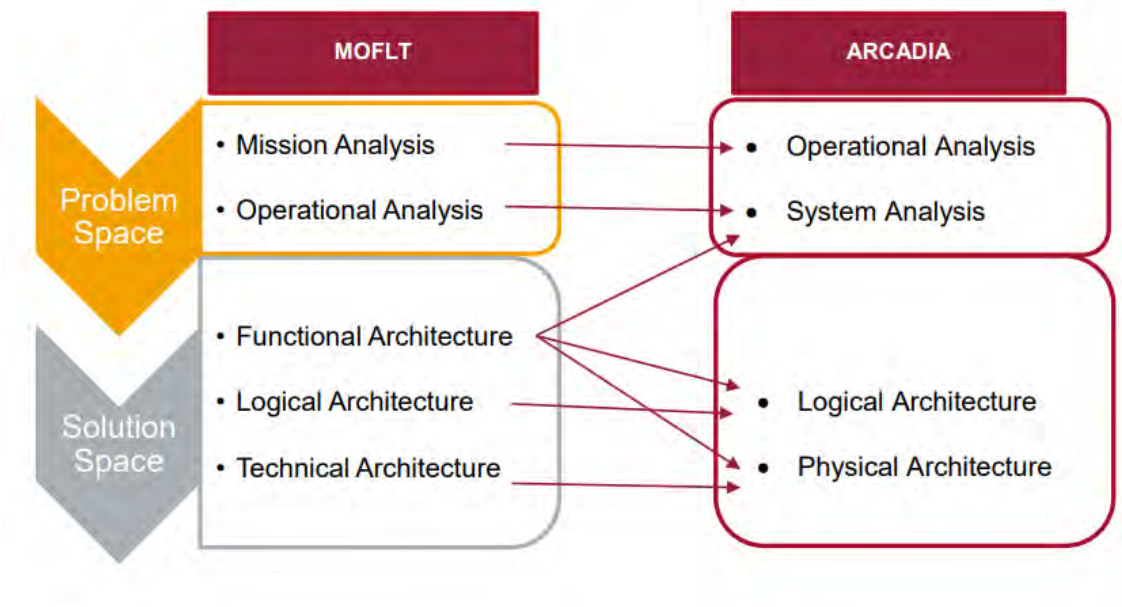

Figura 1.3 Comparación entre MOFLT y ARCADIA..

En el contexto de la progresiva adopción de la Ingeniería de Sistemas Basada en Modelos (MBSE) en el sector espacial europeo, la necesidad de coordinar los esfuerzos y fomentar la colaboración entre los distintos actores se ha vuelto primordial. Respondiendo a este imperativo, surge la iniciativa *"Model Based for System Engineering"* (MB4SE), un esfuerzo concertado para federar a la comunidad de ingeniería de sistemas espaciales y trabajar conjuntamente hacia el despliegue efectivo de la Ingeniería Basada en Modelos (MBE) en los proyectos espaciales europeos.

Previamente a MB4SE, diversas iniciativas tanto industriales como en el marco de los programas de Investigación y Desarrollo (I+D) de la Agencia Espacial Europea (ESA) ya apuntaban hacia esta visión. No obstante, se identificó una carencia de coordinación que limitaba la consecución de sinergias y la convergencia hacia objetivos compartidos. En este escenario, se propuso la federación de dichas iniciativas bajo el marco común de MB4SE for Space, con el fin de optimizar los recursos y alinear las estrategias. Un elemento central de la iniciativa MB4SE es su enfoque metodológico, el cual se fundamenta en el desarrollo y la utilización de una base semántica de referencia, concretada en una Ontología de Sistemas Espaciales (*Space System Ontology*), y en el concepto de un centro de modelos común (*common model hub*). Estos componentes son cruciales para asegurar la interoperabilidad y la coherencia en el intercambio de información entre modelos. De esta manera, la iniciativa MB4SE for Space se constituye como una plataforma estratégica para el diálogo técnico entre los diferentes interesados (*stakeholders*). En este foro, se facilita la discusión y la formulación de recomendaciones clave orientadas a la materialización y consolidación tanto de la Ontología de Sistemas Espaciales como del mencionado centro de modelos.[European Space Agency, sf]

1.3 Objetivos del Trabajo de Fin de Grado

El presente Trabajo de Fin de Grado (TFG) tiene como propósito fundamental complementar la formación teórica en ingeniería con práctica, inmersa en el contexto industrial actual. Se reconoce que la aplicación real de los conocimientos y el contacto directo con los desafíos y metodologías empleadas en la industria constituyen un valor añadido insustituible para la formación del ingeniero. Por ello, se ha optado por fundamentar este trabajo en las actividades y conocimientos adquiridos durante un periodo de prácticas extracurriculares, lo que ha permitido el acceso a información

técnica especializada y el manejo de herramientas y procesos no disponibles en el ámbito puramente académico o mediante la consulta de fuentes públicas.

Ante el creciente uso de recopiladores y sintetizadores de internet, también conocidos como "Chatbot de inteligencia artificial generativa", los TFGs que se limiten a leer artículos en revistas, resumirlos y combinarlo todo junto bajo un mismo texto tendrán valor nulo. Es por ello que el autor ha procurado “mancharse las manos”, es decir, usar programas informáticos no al alcance de la IA. En consecuencia, este trabajo se ha enfocado en la aplicación práctica de la Ingeniería de Sistemas Basada en Modelos (MBSE), utilizando software y metodologías específicas del sector, con el fin de desarrollar un análisis profundo y original que trascienda la mera recopilación bibliográfica.

De forma más concreta, los objetivos que se persiguen con este TFG son los siguientes:

1. Comprender en profundidad los principios, metodologías y herramientas que conforman la Ingeniería de Sistemas Basada en Modelos (MBSE).
2. Analizar la aplicación actual de MBSE en la industria, mediante la revisión de la literatura técnica y el estudio de casos de uso relevantes.
3. Participar y documentar un caso práctico de aplicación de MBSE en un entorno industrial real, derivado de la experiencia obtenida durante las prácticas profesionales.
4. Evaluar y argumentar la pertinencia y los beneficios de promover la adopción de MBSE en sectores de alta complejidad, con especial atención en la industria aeroespacial.
5. Discutir las tendencias actuales, la proyección a futuro de MBSE y las posibles áreas de interés para desarrollos o investigaciones posteriores.

1.4 Estructura del Documento

El TFG se ha planteado de la siguiente manera.

- Una introducción que contextualiza rápidamente la materia y capta el interés del lector de lo que significa MBSE y para qué sirve. Una manera eficiente en el mundo de la inmediatez para atraer la lectura y el interés. Se explicarán la motivación de este TFG y el contexto formativo en el que se inscribe. Se corresponde con el punto 1 del índice.
- Un desarrollo más prolijo de conceptos como: Ingeniería de Sistemas, Modelos, MBSE, Metodología, Herramientas,... Así como su exposición con casos prácticos y tratando de mantener el contacto siempre con la industria y con las tecnologías en el presente en marcha utilizadas por las empresas. A su vez se compararán con la tradicional forma de hacer ingeniería de sistemas basada en documentos y se explicarán las ventajas e inconvenientes enfrentándose entre sí.
- Se comentará qué es SysML, de dónde viene, cómo Arcadia recoge y adapta este lenguaje para resolver problemas ingenieriles. Se hablará de distintas herramientas como Cameo, Capella,... Analizando la utilidad, el campo de cada una de ellas y la metodología con la que suelen venir asociadas.
- A continuación, se centrará en Arcadia y en Capella, debido a que es el lenguaje que se ha facilitado en las prácticas extracurriculares para aprender. Por ello, se explicará más a fondo la metodología usada para el modelado y el software utilizado. Más adelante se intentará ver cómo se integran estas tecnologías con herramientas de simulación, CAD y otras plataformas utilizadas en las fases iniciales del diseño. A toda esta plataforma colaborativa se la llama PLM.
- Se intentará realizar un caso de modelado aplicado a un sistema aeronáutico, en principio el sistema de presurización y aire acondicionado del C-295. Se corresponde con el punto 3.

- Finalmente se discutirá lo aprendido en el proyecto. Se corresponde con el punto 4.
- Y se terminará con las posibles vías de investigación futuras. Se comentará el futuro de la tecnología. Se corresponde con el punto 5.

2 Marco Teórico: Fundamentos de Ingeniería de Sistemas y MBSE

2.1 Introducción a la Ingeniería de Sistemas (IS)

La ingeniería de sistemas se trata de una disciplina interdisciplinaria que se enfoca en el diseño, la integración y la gestión de sistemas complejos a lo largo de todo su ciclo de vida. Su objetivo principal es asegurar que todas los ingenieros que en cualquier problema de un sistema trabajen juntas de manera cohesiva y eficiente para cumplir con los requisitos y objetivos establecidos, equilibrando diversos factores como el coste, el cronograma, el rendimiento y el riesgo. Utiliza procesos estructurados y metodologías para abordar la complejidad inherente al desarrollo de dichos sistemas.

Los problemas que la ingeniería de sistemas busca resolver raramente son puramente técnicos. A menudo tienen dimensiones económicas, sociales, políticas y éticas. Un enfoque interdisciplinario permite considerar todas estas facetas para llegar a soluciones más robustas y sostenibles. De ahí su interdisciplinariedad.

2.1.1 Orígenes y Evolución

Si bien establecer con precisión el génesis de la ingeniería de sistemas resulta complejo, existe un consenso general que sitúa sus inicios formales en la etapa que continúa a la Segunda Guerra Mundial, vinculada al desarrollo de sistemas militares de gran envergadura, como los misiles.

Aunque cabe rastrear antecedentes de esta disciplina en épocas anteriores. Antes de su consolidación formal existía un rol análogo al del ingeniero de sistemas contemporáneo en la ejecución de proyectos de gran magnitud, ejemplos de los cuales incluyen las pirámides de Egipto, los acueductos romanos, la presa Hoover, el puente Golden Gate o el edificio Empire State, predominantemente en el ámbito de la ingeniería civil. Estos profesionales precursores llevaban a cabo sus labores sin el respaldo de una teoría formalizada o una ciencia de la ingeniería de sistemas, y carecían de procesos o prácticas definidas y estandarizadas. Sin embargo se puede buscar su conexión con la ingeniería de sistemas por trabajar con proyectos y problemas enormes, en los que era necesario tener una visión integral y holística del proceso de su construcción... Otro ejemplo se encuentra durante la Segunda Guerra Mundial, la gestión de proyectos como el desarrollo de programas aeronáuticos podía ser supervisada por un director de proyecto o ingeniero jefe, asistido por responsables de los subsistemas fundamentales (propulsión, controles, estructura, entre otros). Fue en este periodo bélico y en la inmediata posguerra cuando ciertos componentes adicionales de la ingeniería de sistemas, destacando la investigación de operaciones y el análisis de decisiones, adquirieron una relevancia creciente.

La acuñación del término "ingeniería de sistemas" se atribuye a Bell Laboratories durante la década de 1940. Posteriormente, en 1950, G. W. Gilman, en el Instituto Tecnológico de Massachusetts (MIT), llevó a cabo lo que se considera la primera iniciativa para impartir formalmente la ingeniería de sistemas. Hacia finales de la década de 1950, la ingeniería de sistemas empezó a configurarse como una disciplina de ingeniería diferenciada. Este proceso de consolidación fue notablemente estimulado por la carrera espacial y la pugna por el desarrollo de misiles nucleares, elementos que se percibían como cruciales para la seguridad nacional de la época. Las marcadas competencias entre las distintas ramas de las fuerzas armadas estadounidenses (Ejército, Marina y Fuerza Aérea) por crear sistemas de alta fiabilidad y asegurar una posición de liderazgo en su implementación y manejo, también contribuyeron a este avance. En este contexto de alta competitividad, tanto las entidades militares como sus principales empresas contratistas (entre ellas, Boeing, Lockheed y Rockwell) se abocaron a la búsqueda de herramientas y metodologías que les permitieran optimizar el desempeño de los sistemas (logro de la misión) y la gestión integral de los proyectos (incluyendo el rendimiento técnico, la adhesión a los cronogramas y el control de los costes).

De este escenario emergió la herramienta PERT (Program Evaluation & Review Technique), una metodología de planificación de naturaleza cuasi-estadística, concebida para mejorar las estimaciones de los plazos de conclusión en proyectos caracterizados por múltiples tareas interconectadas, ya sea de forma secuencial, paralela o interdependiente. La técnica PERT ofrece una perspectiva clara sobre las posibles repercusiones que los adelantos o demoras en tareas específicas pueden tener sobre la fecha final de entrega. Un ejemplo destacado de su aplicación fue por parte de la Marina de los Estados Unidos, que empleó PERT con resultados favorables en el programa de desarrollo del misil Polaris A1, logrando efectuar el primer ensayo de lanzamiento en un lapso de 18 meses desde el comienzo del programa.

En el ámbito académico, en 1957, Goode y Machol, de la Universidad de Michigan, aportaron la publicación "Systems Engineering", en la cual identificaron la emergencia de un paradigma de pensamiento sistémico y metodologías aplicadas al diseño de equipamiento. Simultáneamente, la disciplina de la ingeniería de sistemas experimentaba un desarrollo análogo en el sector empresarial. Arthur Hall, basándose en su trayectoria en el sector de las telecomunicaciones en AT&T, contribuyó con una de las primeras obras monográficas sobre Ingeniería de Sistemas en el año 1962.

En esta misma etapa, la gestión en ingeniería experimentó una evolución que llevó a la normalización del empleo de especificaciones técnicas, documentos para el control de las interfaces, revisiones formales de diseño y mecanismos para un control de cambios riguroso. La irrupción de las computadoras, tanto híbridas como digitales, supuso un avance crucial al posibilitar la realización de simulaciones y evaluaciones exhaustivas de sistemas, subsistemas y componentes individuales. Esto, a su vez, facilitó una síntesis más precisa de los diversos elementos que componen un sistema y permitió efectuar análisis de trade-offs (o soluciones de compromiso) en el diseño de manera más informada. El aprendizaje derivado de las dificultades y los fracasos fue considerable, impulsando innovaciones a lo largo de todas las etapas del desarrollo de productos de avanzada tecnología. Estas innovaciones abarcaron desde la ingeniería y los procesos de adquisición, hasta la fabricación, las pruebas y el control de calidad, con la fiabilidad del sistema como uno de los principales motores de dicho progreso.

Entre las transformaciones e innovaciones más significativas introducidas durante este periodo, destacan:

- **Trazabilidad de componentes:** Ante la variabilidad en la calidad de piezas idénticas provenientes de distintos suministradores o incluso de diferentes lotes de un mismo proveedor, se implementaron sistemas para la identificación unívoca de cada pieza por proveedor y lote. Esto permitía su seguimiento hasta su ubicación final en el ensamblaje, facilitando su sustitución en caso de detectar defectos.

- **Control de materiales y procesos:** Se constató que ciertos acabados o adhesivos, por ejemplo en placas de circuitos, podrían degradarse bajo condiciones extremas como las variaciones térmicas en el vacío espacial tras múltiples ciclos orbitales. Consecuentemente, se hizo imperativo definir, especificar, ensayar y verificar meticulosamente tanto los materiales como los procesos empleados por todos los proveedores.
- **Gestión formal de cambios:** La práctica de modificar diseños o procesos de manera informal con la intención de mejorar el producto, sin la debida actualización de la documentación técnica, a menudo dificultaba la identificación de las causas raíz de los fallos. Para solventar esto, se establecieron procedimientos más rigurosos que exigían la documentación, revisión y aprobación previa de cualquier modificación, lo que llevó a la creación de comités formales de control de cambios en numerosas organizaciones.
- **Optimización de la responsabilidad sobre el producto:** Los esquemas de producción masiva, donde cada operario se centraba en tareas muy específicas, diluían la responsabilidad individual sobre productos de alto valor añadido. La no disponibilidad de componentes críticos, software o la no ejecución de pruebas en los momentos oportunos ocasionaba onerosas demoras. Como respuesta, se instituyeron figuras como los gestores de producto y supervisores específicos (denominados "bird watchers" en un programa de misiles) para garantizar la provisión de todas las partes necesarias y la correcta realización de los ensayos.
- **Control formalizado:** La ausencia de una definición temprana y un control riguroso de las interconexiones entre los distintos componentes, subsistemas y elementos del sistema conllevaba que, aunque cada elemento individual fuese funcional, el conjunto no operase correctamente. Esta situación generaba serios problemas durante las fases de integración, provocando en ocasiones importantes retrasos e incluso la cancelación de programas.

En etapas subsiguientes, personalidades como Sage, Blanchard & Fabrycky, y Boonton & Ramo, en su mayoría con formación en ingeniería, se erigieron como figuras prominentes en la consolidación del campo. El reconocimiento institucional de la disciplina se materializó tempranamente; por ejemplo, en 1966, el Departamento de Defensa de los Estados Unidos admitió la necesidad de directrices específicas para la ingeniería de sistemas, comenzando la publicación de estándares que detallaban su aplicación en el contexto de diferentes sistemas de defensa [Weigel, 2000].

2.1.2 Principios Fundamentales

La ingeniería de sistemas se sustenta en un conjunto de principios que orientan su aplicación en la resolución de problemas complejos y el desarrollo de sistemas eficaces. Estos principios incluyen.

- **Enfoque Interdisciplinario y Holístico:** La ingeniería de sistemas se define como un enfoque interdisciplinario y completo para la resolución de problemas complejos. Adopta una perspectiva holística para descomponer problemas intrincados en partes manejables y luego reintegrar estas partes en una solución cohesiva final. Esta disciplina amalgama elementos de múltiples campos del saber, tales como la investigación de operaciones, el modelado y simulación de sistemas, el análisis de decisiones, la gestión de proyectos y el control, entre otros. Se considera una disciplina global que gestiona los compromisos (trade-offs) y la integración entre los distintos elementos de un sistema para conseguir el mejor producto y/o servicio global.
- **Orientación a los Requisitos y Objetivos del Interesado:** Un pilar fundamental es la satisfacción de los requisitos de los interesados (stakeholders). Esto conlleva la identificación y cuantificación de los objetivos del sistema, así como la transformación de una necesidad operativa identificada en una descripción detallada de los parámetros de rendimiento del sistema y una configuración preferente para el mismo.

- **Proceso Iterativo:** La ingeniería de sistemas se caracteriza por ser un proceso iterativo. Este proceso cíclico abarca fases como el análisis funcional, la síntesis de soluciones, la optimización, la definición detallada, el diseño, la realización de pruebas y la evaluación continua para desarrollar la configuración del sistema más adecuada. Por ejemplo, se describe un proceso de seis fases que incluye la identificación y cuantificación de metas, la creación de conceptos alternativos, la ejecución de análisis de viabilidad y compromiso (trade-offs), la selección e implementación del diseño óptimo, la verificación de la correcta construcción e integración, y una evaluación posterior a la implementación.
- **Descomposición, Síntesis e Integración:** La metodología emplea un enfoque inicialmente reduccionista para desglosar los requisitos complejos del sistema en componentes más pequeños y manejables. Posteriormente, se procede a integrar la implementación física de dichos componentes. Este proceso incluye la integración de parámetros técnicos afines y la garantía de compatibilidad entre todas las interfaces, ya sean físicas, funcionales o programáticas, con el fin de optimizar la definición y el diseño integral del sistema. Este abordaje holístico de descomposición y reintegración es crucial para gestionar la complejidad inherente a los sistemas.
- **Análisis de Alternativas y Optimización (Trade-offs):** Este principio implica la generación de múltiples conceptos de diseño alternativos para el sistema. Se realizan análisis comparativos y de compromiso (design trades) entre estas alternativas para seleccionar e implementar el diseño que mejor satisfaga los objetivos establecidos. La ingeniería de sistemas es, en este sentido, una disciplina eminentemente cuantitativa que involucra la evaluación de trade-offs, la optimización de soluciones, la selección informada y la integración coherente de los productos de diversas disciplinas ingenieriles.
- **Verificación y Validación:** Se lleva a cabo una verificación exhaustiva para asegurar que el diseño del sistema se ha construido e integrado de manera correcta. Adicionalmente, se realiza una evaluación posterior a la implementación para determinar en qué medida el sistema cumple con los objetivos para los que fue concebido. Tanto las pruebas como la evaluación forman parte integral del proceso iterativo de desarrollo.
- **Gestión del Proceso y Enfoque Metodológico:** La ingeniería de sistemas se fundamenta en la aplicación de un proceso bien definido y estructurado, el cual actúa como el nexo común de la disciplina a través de sus diversas áreas de aplicación. Dicho proceso engloba la definición de requisitos, la creación de conceptos alternativos, el análisis de estos conceptos y la realización de estudios de compromiso (trades), la selección del diseño final y, en última instancia, el ensamblaje, la verificación y el despliegue del sistema físico. Se considera, por tanto, que la ingeniería de sistemas es simultáneamente un proceso de diseño y una filosofía de gestión.

2.1.3 Apoyo de la Ingeniería de Sistemas al Ciclo de Vida

La ingeniería de sistemas ofrece un marco metodológico robusto para el desarrollo, la operación y el sostenimiento de los sistemas a lo largo de la totalidad de su ciclo de vida:

- **Integración de Especialidades del Ciclo de Vida:** Desde las fases más tempranas del desarrollo, la ingeniería de sistemas se encarga de integrar consideraciones cruciales como el rendimiento, la producibilidad, la fiabilidad, la mantenibilidad, la operabilidad por humanos ("manability") y, de forma destacada, la soportabilidad (supportability) dentro del esfuerzo global de ingeniería.
- **Consideración de Fases Posteriores y Evolución del Sistema:** El proceso de ingeniería de sistemas contempla una evaluación posterior a la implementación (post-implementation assessment). Esta fase es crítica para determinar el grado de cumplimiento de los objetivos

originales por parte del sistema, proporcionando una retroalimentación valiosa para futuros ciclos de desarrollo, la implementación de mejoras o la gestión de la evolución natural del sistema a lo largo del tiempo.

- **Establecimiento de Prácticas para la Sostenibilidad:** La experiencia acumulada y las lecciones aprendidas a partir de dificultades y fallos en el desarrollo de sistemas complejos han catalizado la evolución de prácticas específicas, intrínsecas a la ingeniería de sistemas, que son fundamentales para el soporte efectivo del ciclo de vida. Entre estas prácticas se incluyen la trazabilidad de piezas, el control riguroso de materiales y procesos, y un sistema formalizado para el control de cambios. También son relevantes la mejora en la rendición de cuentas del producto y el control formal de interfaces.
- **Adaptabilidad a Través de la Iteración:** El carácter inherentemente iterativo de la ingeniería de sistemas facilita la adaptación continua y la gestión proactiva de cambios. Esto permite al sistema evolucionar a lo largo de las distintas fases de su ciclo de vida, ya sea para incorporar nuevas tecnologías, responder a modificaciones en los requisitos iniciales o ajustarse a cambios en el entorno operativo.
- **Visión Integral y Aplicación Multinivel:** La ingeniería de sistemas se aplica a sistemas en una amplia gama de niveles de integración y complejidad. Esta perspectiva abarca desde la concepción inicial y la definición de requisitos, pasando por las fases de diseño, desarrollo, producción, operación y mantenimiento, hasta la eventual retirada o desmantelamiento del sistema, asegurando un enfoque coherente y un soporte continuo durante toda su existencia.

2.2 Ingeniería de Sistemas Basada en Modelos (MBSE)

Dentro de este gran campo que por definición es difícilmente delimitable, se puede encontrar el MBSE. El objetivo del presente trabajo. Por empezar por lo que dice la principal institución en materia de Ingeniería de sistemas. La definición que nos ofrece la INCOSE [INCOSE, 2023] es la siguiente:

> "La ingeniería de sistemas basada en modelos (MBSE) es la aplicación formalizada del modelado para respaldar los requisitos del sistema, el diseño, el análisis, las actividades de verificación y validación que comienzan en la fase de diseño conceptual y continúan durante todo el desarrollo y las fases posteriores del ciclo de vida"

INCOSE SE Vision 2020 (INCOSE-TP-2004-004-02, Sep 2007)

Ahora bien, ¿qué es un modelo? Para tener un primer aproximamiento se verá lo que dice la palabra misma. Se irá a la etimología de la palabra.

La palabra "modelo" tiene un origen interesante que se remonta al latín. El termino llega al español a través del italiano modello, un término que se popularizó durante el Renacimiento. A su vez, modello deriva del latín modulus, que significa "medida" o "pequeña medida". Este término latino es un diminutivo de modus, que se traduce como "manera", "modo" o "medida". Por lo tanto, en su raíz, la palabra "modelo" está ligada a la idea de algo que sirve como medida o referencia.

Esta medida fue evolucionando a algo más complejo. Un conjunto de normas o reglas que un artista debía seguir para realizar una obra, este artista se basaba en un modelo. Recogiendo diversas acepciones comunmente utilizadas se puede obtener la siguiente definición, que es bastante amplia y permite entender el conjunto de casos que abarca, lo que en un origen fue una simple medida.

Se define por modelo:

- Una versión simplificada de un concepto, fenómeno, relación, estructura o sistema.
- Una representación gráfica, matemática o física.

- Una abstracción de la realidad eliminando componentes innecesarios.
- Los objetivos de un modelo incluyen:
  - Facilitar la comprensión.
  - Ayudar en la toma de decisiones, examinar escenarios hipotéticos.
  - Para explicar, controlar y predecir eventos.

¿Cuál es la utilidad que puede tener el utilizar modelos para resolver problemas en la realidad? Se pondrá de ejemplo para explicar la idea de modelado el diseño de la nave espacial Nexus, un precursor del Telescopio Espacial James Webb. De esta manera se verá cómo se puede utilizar un modelo para tomar e influir en decisiones del diseño de un producto. Se detallarán dos configuraciones de Nexus: una almacenada y otra desplegada.

El primer paso pasa por modelar la física de los subsistemas. Para la estructura ligera y flexible del telescopio, se desarrolla un modelo de elementos finitos (FEM) que predice la masa total, la matriz de inercia y, fundamentalmente, los modos naturales de vibración y las formas modales bajo diferentes excitaciones (incluyendo el parasol, los paneles solares y el telescopio óptico). A partir de este FEM, se crea un modelo dinámico más integrado usando MATLAB y Simulink. Este modelo de Simulink incluye el FEM, diversas fuerzas y fuentes de ruido (ruido de ruedas de reacción, criorefrigerador) y los actuadores del sistema de control de actitud.

El modelo predice el rendimiento del telescopio en base a dos métricas principales: el "jitter" de la línea de visión (LOS) y el error del frente de onda (distorsión óptica). El modelo completo tiene 2 parámetros de rendimiento, 25 parámetros de diseño, 320 estados y 4 fuentes de perturbación. Al simular el rendimiento, se observa (mediante diagramas de Bode y trazas temporales del centroide de la imagen) que el "jitter" de la línea de visión (15 micras RMS) no cumple el requisito de 5 micras.

Para comprobar esto, se va a analizar la sensibilidad del rendimiento a los parámetros de diseño (usando la matriz Jacobiana). Se identifican como parámetros críticos el desequilibrio dinámico (UD) de las ruedas de reacción y la rigidez del aislador (K_rISO) en el que se montan. Mapeando el rendimiento en función de estos dos parámetros, se pueden visualizar contornos de igual rendimiento. El diseño inicial no cumple los requisitos, por lo tanto se exploran diferentes acciones:

- Hacer el aislamiento mucho más suave manteniendo la perturbación.
- Reducir la perturbación (ruedas de reacción muy silenciosas y bien equilibradas), como hizo el Hubble.
- Un enfoque equilibrado: suavizar el aislador y reducir el ruido original.

Así el modelo ayuda a identificar las causas del bajo rendimiento y a explorar diferentes estrategias de diseño para cumplir con los requisitos.

He aquí la clave, un modelo es un sistema de conceptos que permite ver de manera anticipada los parámetros críticos del diseño que se debe modificar. De esta manera, es posible anticiparse a etapas posteriores en las que se realizarán las pruebas físicas, infinitamente más costosas y complicadas de simular.

Mientras que los modelos de **FEM (Modelo de Elementos Finitos)** se centran en el análisis y la simulación del comportamiento físico detallado de componentes bajo cargas y condiciones específicas mediante la resolución de ecuaciones diferenciales complejas, y los modelos de **Simulink** se especializan en la simulación de sistemas dinámicos y de control, el enfoque se desplaza ahora hacia el **modelado gráfico**. Este último constituye la fase primordial y conceptual de nuestro proyecto, con la que se conseguirá definir la **arquitectura** del problema ingenieril que se abordará en este trabajo.

2.3 Lenguajes de modelado gráfico

Como ya se indicaba en la definición de INCOSE, no se modela de cualquier manera. Se deben seguir reglas, el modelo debe estar formalizado. Una manera de expresar nuestros diagramas de manera clara y sistemática es con lenguajes de modelado gráfico.

2.3.1 SysML: Un Lenguaje de Modelado para Sistemas Complejos

SysML (Systems Modeling Language) es un lenguaje de modelado de propósito general diseñado para la ingeniería de sistemas basada en modelos (MBSE). Su objetivo principal es facilitar la creación y visualización de modelos que representen diversos aspectos de un sistema complejo.

SysML te ayuda a especificar, analizar, diseñar y verificar sistemas complejos que pueden incluir hardware, software, información, personal, procedimientos e instalaciones.

Permite modelar los requisitos, el comportamiento, la estructura y los parámetros de un sistema cuya clasificación se puede observar en la **Figura 2.1**.

SysML facilita la comunicación, el análisis y la colaboración entre los ingenieros de sistemas.

El fundamento del SysML, UML.

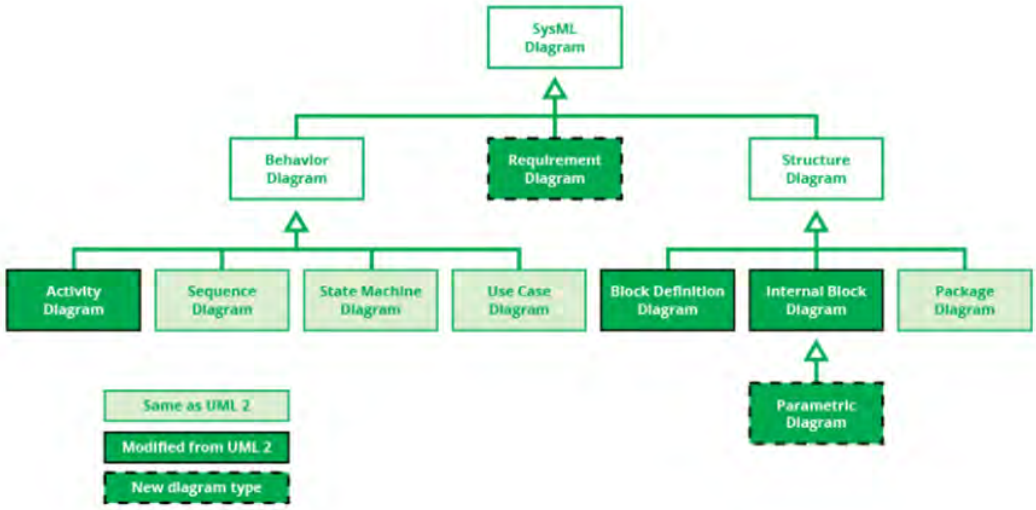

Figura 2.1 Clasificación de los diagramas del Lenguaje de Modelado de Sistemas (SysML), mostrando su estructura y herencia desde UML 2 y UML.

SysML extiende el Lenguaje de Modelado Unificado (UML) al incorporar conceptos adicionales adaptados a la ingeniería de sistemas:

- **Extensión de UML:** SysML se basa en UML, pero se ha modificado para incluir construcciones específicas para la ingeniería de sistemas.
- **Menos centrado en el desarrollo de software:** SysML reduce el enfoque en el software que tiene UML y agrega diagramas para la gestión de requisitos y el análisis de rendimiento.
- **Más pequeño y fácil de aprender:** SysML elimina muchas construcciones centradas en el software, lo que lo hace más conciso que UML.

En la figura de arriba se parte de estos diagramas que son básicos para la ingeniería de Sistemas, presentes tanto en SysML y UML2.

- **Behavior Diagram (Diagrama de Comportamiento):** Se enfoca en describir la funcionalidad dinámica del sistema, es decir, cómo actúa y reacciona.
  - Diagramas de casos de uso
  - Diagrama de estados de máquina
  - Diagramas de secuencia
- **Structure Diagram (Diagrama de Estructura):** Describe la arquitectura estática del sistema, sus componentes y cómo están conectados.
  - **Package Diagram (Diagrama de Paquetes):** Se utiliza para organizar los elementos del modelo en grupos lógicos (paquetes) y mostrar las dependencias entre ellos, al igual que en UML.

En cambio recoge del UML 2, aún habiéndolos modificado:

- **Behavior Diagrams:**
  - **Activity Diagram (Diagrama de Actividad):** SysML toma el diagrama de actividad de UML y lo adapta o extiende para las necesidades de la ingeniería de sistemas.
- **Structure Diagrams:**
  - **Block Definition Diagram (Diagrama de Definición de Bloques)** Es una evolución del diagrama de clases de UML, adaptado para representar bloques (componentes del sistema) y sus relaciones estructurales.
  - **Internal Block Diagram (Diagrama de Bloques Internos):** Deriva del diagrama de estructura compuesta de UML y se utiliza para describir la estructura interna de un bloque, mostrando sus partes, puertos y conectores.

Las adiciones claves en el SysML son:

- **Requirement Diagram:** Es una adición clave de SysML que no existe de forma nativa en UML. Permite modelar requisitos, su descomposición, derivaciones y satisfacción por otros elementos del modelo.
- **Parametric Diagram (Diagrama Paramétrico):** Este es otro diagrama nuevo en SysML, crucial para el análisis de ingeniería, ya que muestra restricciones y relaciones matemáticas (como ecuaciones) entre las propiedades de los bloques. Realmente relevante de cara a simulaciones y optimización.

2.3.2 Diagramas de SysML

SysML utiliza diagramas para representar diferentes aspectos de un sistema. Existen nueve tipos principales de diagramas, algunos de los cuales ya se han explicado por encima. Se pueden clasificar en tres categorías:

Diagrama de Definición de Bloques (BDD - Block Definition Diagram)

Es el diagrama fundamental para describir la arquitectura del sistema. Los "bloques" son las unidades modulares básicas de un sistema y pueden representar componentes de hardware, software, datos, personas o incluso procesos. El BDD no solo muestra estos bloques, sino también sus características (propiedades como partes, referencias, valores y operaciones) y las diversas relaciones entre ellos, como asociaciones (relaciones estructurales generales), generalizaciones (herencia), dependencias y composiciones/agregaciones (relaciones parte-todo). Permite definir un sistema en términos de sus jerarquías y clasificaciones de bloques [Delligatti, 2014].

Diagrama de Bloques Internos (IBD - Internal Block Diagram)

Mientras que el BDD define los tipos de bloques, el IBD se enfoca en la realización interna de un bloque específico. Muestra las "partes" (instancias de otros bloques) que componen el bloque, los "puertos" (puntos de interacción con el exterior) y los "conectores" que unen las partes entre sí o los puertos de las partes con los puertos del bloque encapsulador. El IBD es esencial para entender cómo las diferentes piezas de un bloque colaboran y se comunican para lograr la funcionalidad del bloque. También puede mostrar los flujos de ítems (item flows) entre las partes a través de los conectores [Object Management Group, 2022].

Diagrama de Paquetes (Package Diagram)

Al igual que en UML, este diagrama se utiliza para agrupar elementos del modelo (como bloques, casos de uso, requisitos) en "paquetes", que actúan como contenedores o espacios de nombres. Ayuda a gestionar la complejidad de modelos grandes, dividiéndolos en partes más manejables y mostrando las dependencias entre estos paquetes (por ejemplo, una importación o una dependencia de acceso). Esto facilita la navegación, la comprensión y la reutilización de partes del modelo.

Diagramas de Comportamiento

- **Diagrama de Casos de Uso (Use Case Diagram):** Representa la funcionalidad del sistema desde la perspectiva de un usuario externo o "actor". Cada "caso de uso" describe una interacción que produce un resultado observable y de valor para el actor. Define el alcance de lo que el sistema debe hacer (el "qué", no el "cómo").
- **Diagrama de Actividades (Activity Diagram):** Describe la secuencia de acciones y el flujo de control y de objetos/datos en un comportamiento. Incluye elementos como acciones, nodos de inicio y fin, nodos de decisión, nodos de fusión, nodos de bifurcación para flujos paralelos y nodos de unión. SysML extiende las capacidades de UML para manejar mejor los flujos continuos, la asignación de acciones a partes del sistema (mediante "activity partitions" o swimlanes) y la representación de flujos de energía o materia. Es útil para modelar procesos de negocio, operaciones de software y flujos de trabajo de sistemas complejos.
- **Diagrama de Secuencia (Sequence Diagram):** Se centra en la interacción entre los actores y componentes a lo largo del tiempo. Visualiza el intercambio de mensajes (síncronos, asíncronos, de respuesta) en orden cronológico, de arriba hacia abajo. Las "barras de activación" en las lifelines indican cuándo una parte está activa procesando un mensaje. Los "fragmentos combinados" permiten modelar interacciones complejas como bucles (loop), alternativas (alt), opciones (opt) y ejecuciones paralelas (par). Es excelente para detallar un escenario específico de un caso de uso.
- **Diagrama de Máquina de Estados (State Machine Diagram):** Describe los diferentes estados por los que puede pasar un bloque (o una parte de él) a lo largo de su vida, así como los eventos que provocan las transiciones entre esos estados. Cada transición puede tener una condición de guarda (guard) que debe cumplirse y puede desencadenar una acción. Los estados pueden ser simples o compuestos (conteniendo sub-estados). Es fundamental para modelar el comportamiento reactivo de los sistemas, es decir, cómo responden a eventos internos o externos.

Otros Diagramas (Categoría frecuentemente usada para destacar los diagramas clave añadidos por SysML)

- **Diagrama de Requisitos (Requirement Diagram):** Es una adición distintiva de SysML, diseñada para visualizar los requisitos, sus atributos (como identificador, texto, tipo, riesgo, verificación) y las relaciones entre ellos y con otros elementos del modelo. Las relaciones típicas incluyen contención (para requisitos compuestos), derivación (un requisito se deriva

de otro), satisfacción (un elemento de diseño satisface un requisito), verificación (un caso de prueba verifica un requisito), refinamiento (un requisito se detalla más) y trazabilidad (una relación genérica). Esto es crucial para la trazabilidad de los requisitos a lo largo de todo el ciclo de vida del desarrollo del sistema.

- **Diagrama Paramétrico (Parametric Diagram):** Otro nuevo tipo de diagrama en SysML, utilizado para soportar análisis de ingeniería y cuantitativos. Muestra cómo se aplican restricciones (expresadas como ecuaciones o desigualdades) a las propiedades de valor de los bloques del sistema. Utiliza "constraint blocks" (bloques de restricción) que contienen las ecuaciones, y estas restricciones se aplican a las propiedades (parámetros) de los bloques del sistema dentro del diagrama. Esto permite realizar análisis de rendimiento, fiabilidad, masa, coste y otros análisis cuantitativos, asegurando la consistencia entre diferentes vistas de ingeniería y el modelo general del sistema.

2.4 Metodologías de Modelado

2.4.1 Metodología ARCADIA

Dentro del panorama de la Ingeniería de Sistemas Basada en Modelos (MBSE), existen diversas metodologías que buscan guiar el proceso de diseño y análisis de sistemas complejos. Una de las más relevantes y estructuradas, especialmente en sectores como el **aeroespacial, defensa, transporte y energía**, es **ARCADIA (Architecture Analysis & Design Integrated Approach)**. Su nombre indica su propósito: un análisis de la arquitectura que integra el diseño desde sus fases más tempranas.

ARCADIA es una metodología de ingeniería de sistemas y software **centrada en la definición y validación de arquitecturas** para sistemas complejos. Fue desarrollada por **Thales** y se caracteriza por su **enfoque estructurado y dirigido por modelos**, buscando asegurar la coherencia entre las necesidades operacionales del cliente y la solución técnica final. A diferencia de un lenguaje de modelado como SysML, que proporciona la notación pero no el proceso, **ARCADIA define explícitamente un conjunto de pasos y perspectivas de ingeniería a seguir**.

Una característica distintiva de ARCADIA es su fuerte énfasis en el **análisis funcional** y la **asignación explícita de funciones a componentes** en cada nivel de arquitectura (lógica y física). Asimismo, promueve la **validación temprana** y la justificación de las decisiones arquitectónicas mediante una **trazabilidad rigurosa** entre los diferentes niveles y hacia los requisitos.

Aunque ARCADIA define su propio metamodelo y conceptos, está fuertemente inspirada en estándares como UML y SysML. La metodología se implementa principalmente a través de la herramienta de modelado de código abierto **Capella**, que guía al usuario a través de las distintas fases y proporciona los diagramas y elementos específicos de ARCADIA.

En resumen, ARCADIA se presenta como una metodología **robusta y completa** para abordar el diseño arquitectónico de sistemas complejos. Proporciona un **marco estructurado** que facilita la colaboración, asegura la coherencia y la trazabilidad, y permite una **validación progresiva** de la solución frente a las necesidades iniciales. Su enfoque paso a paso a través de distintas perspectivas la convierte en una opción muy relevante para proyectos que requieren un alto grado de rigor en la definición de la arquitectura.

Conceptos Clave en ARCADIA

El modelado de arquitecturas de sistemas complejos requiere un enfoque robusto y bien definido. ARCADIA, como metodología, proporciona un **conjunto específico de conceptos de modelado** adaptados para guiar este proceso, especialmente en su **flujo de trabajo**. La herramienta SMW, al aplicar estos conceptos, sigue una serie de pasos estructurados para construir un modelo de arquitectura completo y coherente.

Elementos Centrales del Modelado en Arcadia

En el corazón de Arcadia se encuentran varios conceptos clave que permiten describir el sistema desde diferentes perspectivas y niveles de abstracción: La **Función** es la piedra angular, representando una acción, tarea o transformación que el sistema o sus partes realizan. Estas funciones evolucionan a través de las capas de Arcadia, comenzando como Funciones Operacionales (lo que los usuarios hacen), refinándose en Funciones del Sistema (lo que el sistema debe hacer para soportar las operacionales), luego en Funciones Lógicas (cómo el sistema se organiza funcionalmente) y finalmente en Funciones Físicas (cómo se implementan esas funciones en hardware y software). Estas distintas capas funcionales están intrínsecamente vinculadas mediante relaciones de realización, donde una función de una capa más concreta implementa o detalla una función de una capa más abstracta.

Para ejecutar estas funciones, Arcadia define **Componentes**. Estos son los bloques estructurales del sistema. Al igual que las funciones, existen Componentes Lógicos (LC), que son abstracciones de cómo se agrupan las funcionalidades, y Componentes Físicos (PC), que representan las entidades concretas. Los Componentes Físicos pueden ser Nodos, que aloja a otros componentes de comportamiento y les da los recursos necesarios o de Comportamiento que tienen asignadas las funciones que desarrollarán. La relación entre componentes lógicos y físicos también se establece mediante la realización.

La interacción entre componentes se gestiona a través de **Puertos**. Un Puerto es un punto de interacción definido en un componente. Existen Puertos de Flujo, diseñados para el paso de ítems (como datos o materia), y Puertos Estándar, que se utilizan para interacciones basadas en servicios y están tipados por Interfaces.

Una **Interfaz** define el contrato para un Puerto Estándar. Especifica las operaciones que un componente puede proporcionar o requerir, así como los ítems que se intercambian a través de esa interfaz.

El flujo de información, materia o energía entre funciones o componentes se denomina **Intercambio**. Hay Intercambios Funcionales que ocurren entre Funciones, e Intercambios de Componentes que se producen a través de Puertos y Conectores entre componentes.

Para estructurar el análisis funcional desde una perspectiva de alto nivel, Arcadia utiliza el concepto de **Capacidad**. Una Capacidad describe una habilidad o servicio que el sistema (o su entorno operacional) ofrece a sus usuarios o a otros sistemas, denominados **Actores**. Los Actores son entidades externas (humanos, otros sistemas) que interactúan con el sistema en una capa determinada, distinguiéndose entre Actores Operacionales y Actores del Sistema.

Aunque los **Requisitos** no son el elemento central del modelado arquitectónico en Arcadia como lo podrían ser en SysML puro, se gestionan y vinculan estrechamente a los elementos del modelo (como funciones y componentes) mediante "relaciones de satisfacción", asegurando que la arquitectura diseñada cumple con lo especificado por la normativa o por el cliente.

La trazabilidad es fundamental en Arcadia, destacando la **Realización** entre capas (por ejemplo, un Componente Físico realiza un Componente Lógico) y la **Mapeado** de funciones a componentes dentro de una misma capa.

Fases de Arcadia.

Gracias a la herramienta Capella que se explicará en detalle más adelante, se implementan los conceptos de Arcadia a través de un proceso metodológico que guía la construcción del modelo de arquitectura:

1. El punto de partida es la definición de las **Capacidades**. Estas describen los servicios fundamentales que el sistema ofrecerá, es decir, aquello que el sistema o producto es capaz de hacer desde una perspectiva de alto nivel.

2. Estas Capacidades se describen y detallan mediante **Cadenas Funcionales y Escenarios**. Las Cadenas Funcionales ilustran cómo se encadenan las funciones para entregar una capacidad,

mientras que los Escenarios ofrecen ejemplos concretos del uso del sistema en un contexto particular.

3. Con esta base, se procede a la formalización del modelo del sistema en las diferentes fases de Arcadia utilizando los diagramas de arquitectura apropiados para cada capa (Operacional, Sistema, Lógica y Física).
4. Las Cadenas Funcionales y Escenarios revelan las funciones necesarias y sus dependencias. Esto implica definir cómo las funciones y componentes interactúan:
    - Tanto funciones como componentes deben tener Puertos para la interacción.
    - Las funciones se asignan a los componentes responsables de su ejecución.
    - Las funciones se conectan entre sí mediante Intercambios Funcionales a través de sus puertos.
    - Los componentes, a su vez, se conectan mediante Intercambios de Componente a través de sus puertos.
5. Es crucial entender que los Intercambios entre Componentes (especialmente en la fase lógica, donde se conciben como partes o subsistemas) transportan los Intercambios Funcionales entre los componentes de origen y destino. De manera similar, los Intercambios entre Componentes Físicos (en la fase física) transportan los intercambios entre los componentes de comportamiento correspondientes.
6. Se concibe una **Máquina de Estados** del sistema para definir sus modos y estados operativos, así como las transiciones válidas entre ellos. Estas transiciones suelen estar relacionadas con los Intercambios Funcionales que las desencadenan o resultan de ellas.
7. Los Intercambios y Puertos Funcionales no son abstractos; transportan datos concretos, denominados **Ítems de Intercambio (Exchange Items)**, que son referencias a conjuntos de datos específicos.
8. Se debe realizar un mapeado (allocation) exhaustivo de estos datos e intercambios para asegurar la coherencia del flujo de información a través del modelo:
    - Los Intercambios Funcionales se mapean sobre los Intercambios entre Componentes Lógicos.
    - Finalmente, los Intercambios entre Componentes Lógicos (de comportamiento) se transicionan sobre los Intercambios Físicos que ocurren entre Componentes de Implementación (nodos de hardware).
9. Paralelamente, se realiza un mapeado de los Puertos (port allocation):
    - Los Puertos Funcionales se mapean sobre los Puertos de los Componentes Lógicos correspondientes.
    - Los Puertos de los Componentes Lógicos (de comportamiento) se mapean sobre los Puertos de los Componentes Físicos.
10. Se definen las **Interfaces** del sistema, que agrupan y referencian los Ítems de Intercambio, sirviendo como contratos formales para la interacción.
11. Estas Interfaces se mapean sobre los Puertos de los Componentes Físicos (de naturaleza comportamiento) que las proveen o las requieren.
12. Un paso continuo y esencial es añadir las referencias de **Realización** entre fases. Esto significa que todo elemento de una fase anterior de Arcadia debe estar realizado por otro elemento en una fase posterior (más concreta) del método, y esta vinculación se añade manualmente en la herramienta.

El Modelo de Arquitectura Resultante

Al implementar todos estos pasos, el sistema queda descrito de manera integral. El resultado no es solo un conjunto de diagramas de arquitectura, sino también una máquina de estados que define su comportamiento, escenarios y cadenas funcionales que ilustran sus casos de uso a través de las capacidades definidas, y un modelo de datos completo. Este modelo de datos se acompaña de un detallado mapeado y referencias de realización que conectan lógicamente los exchange items, los intercambios funcionales, los intercambios entre componentes, las interfaces y los puertos, completando así la información del sistema y asegurando su trazabilidad y coherencia.

Estructura de la Metodología Arcadia

La metodología Arcadia propone un enfoque estructurado para la ingeniería de sistemas, dividiendo el problema de diseño en distintas perspectivas o niveles de análisis. Esta división facilita la gestión de la complejidad y asegura la coherencia desde las necesidades iniciales hasta la solución técnica. Los dos primeros niveles se centran en definir el problema y las necesidades de los stakeholders, mientras que los dos siguientes se enfocan en definir la solución.

En la siguiente figura 2.2, se ve un recorrido de la aplicación de la metodología pasando por distintas capas desde un nivel alto de abstracción hasta la materialización física y concreta. Se da una Visión general de la metodología, mostrando la transición desde las necesidades (OA, SA) a la solución (LA, PA) a través de las capacidades, funciones y estructura del sistema. En el siguiente apartado se explicarán las distintas capas que la componen.

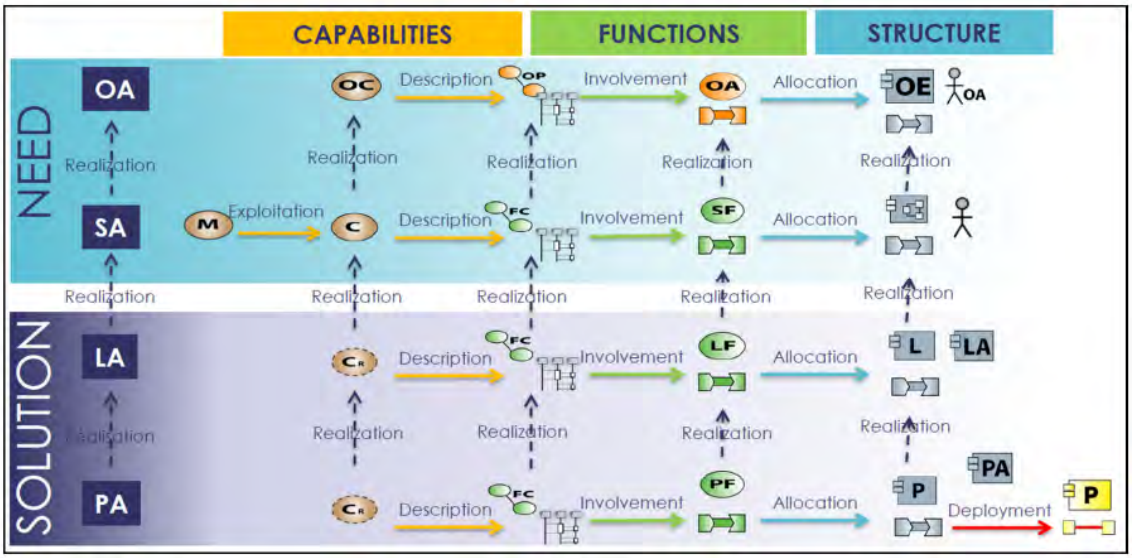

Figura 2.2 Esquema general de la metodología ARCADIA en formato matriz.

1. **Análisis Operacional (OA - Operational Analysis)**
   - **Objetivo Principal:** Comprender el contexto y las necesidades de los usuarios y otros stakeholders, independientemente del sistema a desarrollar. Responde a: "Lo que los usuarios del sistema necesitan conseguir".
   - **Enfoque:** Se analiza el entorno operativo, las actividades que realizan los actores (humanos u organizaciones) para lograr sus metas, y los objetivos o capacidades operacionales que desean alcanzar. Se busca plasmar las necesidades de los actores que interactuarán directa o indirectamente con el futuro sistema, así como las relaciones entre ellos.
   - **Diagramas Comunes en Capella:**

  - **OCD (Operational Context Diagram):** Es el diagrama principal para establecer el contexto operacional. Muestra los Actores Operacionales (entidades externas que interactúan en el ámbito operacional) y sus interacciones de alto nivel, definiendo el perímetro del análisis.
  - **OAB (Operational Architecture Blank / Operational Activity Breakdown Diagram):** Este diagrama sirve como punto de partida para detallar la arquitectura operacional. Puede utilizarse como un "Operational Architecture Blank" para esbozar la estructura general, mostrando las principales Entidades Operacionales y Actores. También se emplea como "Operational Activity Breakdown Diagram" para descomponer las Actividades Operacionales en sub-actividades más manejables, clarificando cómo se alcanzan las metas operacionales.
  - **OES (Operational Entity Scenario):** Escenarios que describen secuencias de interacciones (intercambios operacionales) entre Entidades Operacionales (Actores o el propio sistema operacional) para ilustrar comportamientos específicos o la realización de una capacidad operacional. Ayudan a validar la comprensión de las necesidades y flujos de trabajo.

2. **Análisis Funcional y de Necesidades del Sistema (SA - System Need Analysis)**
   - **Objetivo Principal:** Definir qué debe hacer el sistema para satisfacer las necesidades operacionales identificadas en OA. Responde a: “Lo que el sistema tiene que lograr para los usuarios”.
   - **Enfoque:** Se transforman las necesidades operacionales en funciones de alto nivel que el sistema debe realizar. Se definen los actores que interactuarán directamente con el sistema, las capacidades que este ofrecerá y se establecen los requisitos funcionales y no funcionales principales que debe cumplir. Se delimita la frontera del sistema y las capacidades que este se planea provea.
   - **Diagramas Comunes en Capella:**
     - **MCB (Missions and Capabilities Blank):** Diagrama para definir y visualizar las misiones que los usuarios desean llevar a cabo y las capacidades que el sistema debe proporcionar para soportar dichas misiones.
     - **CSA (Contextual System Actors Diagram):** Muestra el sistema como una caja negra, identificando los Actores del Sistema que interactúan directamente con él.
     - **SAB (System Architecture Blank):** Es el diagrama principal de la perspectiva de Análisis de Sistema.
     - **ES (Exchange Scenario Diagram):** Describe secuencias de interacciones (intercambios funcionales) entre los Actores del Sistema y el Sistema.
     - **SFBD (System Functions Breakdown Diagram):** Diagrama utilizado para descomponer jerárquicamente las Funciones del Sistema.
     - **SDFB (System Data Flow Blank):** Se enfoca en visualizar los flujos de datos entre las Funciones del Sistema.
     - **SFCD (System Functional Chain Diagram):** Representa cadenas funcionales a nivel de sistema.

3. **Arquitectura Lógica (LA - Logical Architecture)**
   - **Objetivo Principal:** Proponer una primera solución conceptual y funcional, independiente de la tecnología de implementación. Responde a: “Cómo el sistema trabajará para cumplir con las expectativas”.

- **Enfoque:** Se detallan las funciones del sistema, asignándolas a componentes lógicos abstractos.
- **Diagramas Comunes en Capella:**
  - **LCBD (Logical Components Breakdown Diagram):** Diagrama de Descomposición de Componentes Lógicos.
  - **LAB (Logical Architecture Blank Diagram):** Diagrama principal de la arquitectura lógica.
  - **LFBD (Logical Functions Breakdown Diagram):** Diagrama de Descomposición de Funciones Lógicas.
  - **LDFB (Logical Data Flow Diagram):** Diagrama de Flujo de Datos Lógicos.
  - **LFCD (Logical Function Context Diagram):** Diagrama de Contexto de Funciones Lógicas.
  - **CDB (Class Diagram Blank):** Se utilizan para modelar la estructura de los datos.

4. **Arquitectura Física (PA - Physical Architecture)**
   - **Objetivo Principal:** Definir la solución técnica concreta. Responde a: "Cómo se desarrollará y construirá el sistema".
   - **Enfoque:** Se seleccionan los componentes físicos (hardware, software) y se define cómo se despliega el software sobre el hardware.
   - **Diagramas Comunes en Capella:**
     - **PCBD (Physical Components Breakdown Diagram):** Diagrama de Descomposición de Componentes Físicos.
     - **PAB (Physical Architecture Blank Diagram):** Diagrama principal de la arquitectura física.
     - **PFBD (Physical Functions Breakdown Diagram):** Diagrama de Descomposición de Funciones Físicas.
     - **PDFD (Physical Data Flow Diagram):** Diagrama de Flujo de Datos Físicos.
     - **Diagramas de Despliegue (Deployment Diagrams):** Representan la asignación de artefactos de software a nodos de hardware.
5. **Diagramas Transversales (Comunes a varias capas)**
   - **FCD (Functional Chain Diagram):** Diagramas de Cadenas Funcionales.
   - **CD (Class Diagram):** Diagramas de Clases.
   - **M&S (Mode and State Machine Diagram):** Diagramas de Máquinas de Estado y Modos.
   - **ID (Interface Diagram):** Diagramas de Interfaces.

Metodología de Modelado Arquitectónico con Arcadia

Esta sección resume la metodología Arcadia aplicada, detallando las fases de Análisis Operacional, Análisis Funcional, Diseño Lógico y Diseño Físico, con sus artefactos clave según SAM-SMW [Roques, 2018]. El flujo de trabajo a través de estas distintas capas de abstracción se ilustra de forma esquemática en la **Figura 2.3**.

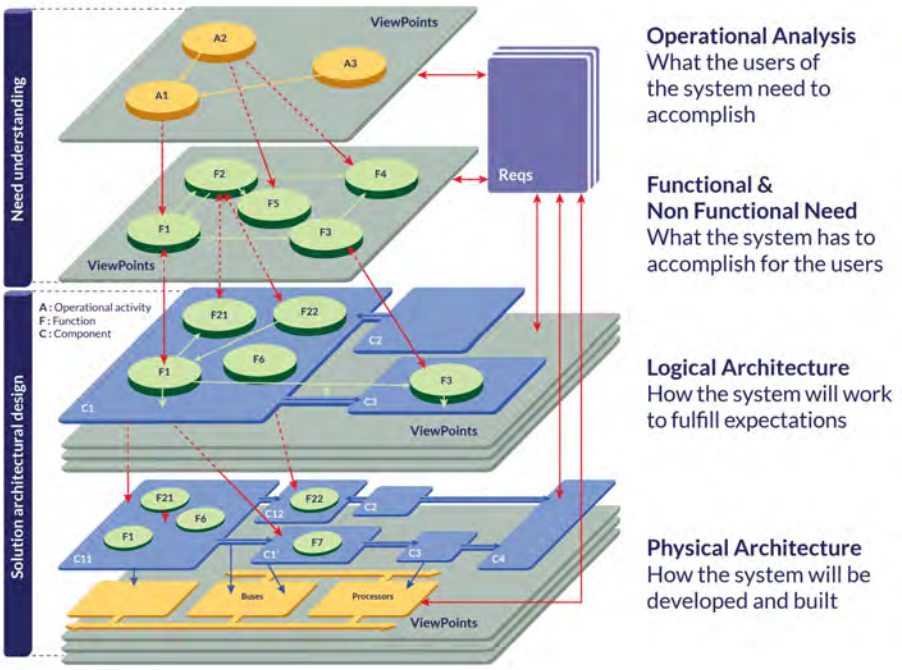

Figura 2.3 Esquema de las capas de la metodología ARCADIA..

1 **Metodología para el Análisis Operacional del Sistema** El análisis operacional se centra en las necesidades del usuario y el contexto operativo.

- Elaborar el Diagrama de Capacidades Operacionales ([OCB]): Identifica los servicios esperados por los actores.
- Desarrollar el Diagrama de Arquitectura Operacional ([OAB]): Describe actividades e interacciones de usuarios y stakeholders.
- Crear Escenarios Operacionales ([OES]): Ilustra el uso de capacidades operacionales en contextos específicos.
- Definir Procesos Operacionales ([OPD]): Documenta secuencias de actividades operacionales relacionadas con capacidades.
- Opcionalmente, se pueden crear diagramas de desglose e interacción ([OABD], [OAIB], [OEBD]) para mayor detalle.

2 **Metodología para el Análisis Funcional del Sistema** Esta fase traduce necesidades operacionales en requisitos funcionales del sistema.

- Transición Operacional a Sistémica: Adapta actores y entidades operacionales al contexto del sistema.
- Elaborar Diagrama de Misiones y Capacidades del Sistema ([MCB]): Define misiones y capacidades sistémicas.
- Desarrollar Diagrama de Contexto del Sistema ([CSA]): Muestra el sistema interactuando con actores externos.
- Crear Arquitecturas Parciales por Caso de Uso ([SAB]): Refina funciones sobre actores, entidades y el sistema para casos de uso.
- Desarrollar Escenarios del Sistema ([ES]): Detalla interacciones funcionales para cada capacidad sistémica.

- Análisis de Funciones del Sistema ([SFBD], [SDFB]): Ordena, categoriza y agrupa funciones, aplicando técnicas de afinidad.
- Definición de Cadenas Funcionales del Sistema ([SFCD]): Muestra secuencias de funciones para realizar capacidades.
- Crear Diagrama de Arquitectura Final del Sistema ([SAB]): Consolida los requisitos funcionales en una vista arquitectónica global.

Se aplica la técnica de "Buceo y Superficie": análisis detallado con escenarios y arquitecturas parciales ([SAB]/[ES]), seguido de una síntesis de alto nivel. Para sistemas complejos, se recomienda el estudio detallado de casos de uso.

3 **Metodología para el Desarrollo de la Fase Lógica** Define la arquitectura conceptual del sistema, abstrayendo la implementación física.

- Transición desde Análisis Funcional: Adapta funciones, actores y capacidades a la perspectiva lógica.
- Elaborar Diagrama de Desglose de Componentes Lógicos ([LCBD]): Estructura los componentes lógicos.
- Desarrollar Diagrama de Arquitectura Lógica ([LAB]): Define componentes lógicos e intercambios entre ellos (CEs).
- Refinar Diagrama de Desglose de Funciones Lógicas ([LFBD]): Detalla o redefine funciones a nivel lógico.
- Construir Diagrama de Arquitectura Lógica Completo ([LAB]): Mapea funciones lógicas (y sus FEs) sobre componentes lógicos (y sus CEs).
- Mapear Intercambios Funcionales a Intercambios de Componentes: Vincula FEs con los CEs que los transportan.
- Opcional: Usar Diagramas de Flujo de Datos Lógicos ([LDFB]): Para navegación y explicación contextual.
- Mapear Funciones del Sistema: Asegura la cobertura de las funciones sistémicas en la lógica.
- Crear Binomios [Arquitectura Lógica Parcial – Escenario Lógico] ([LAB]-[ES]): Provee vistas de casos de uso lógicos.
- Desarrollar Máquina de Estados ([M&S]): Modela el comportamiento dinámico del sistema.
- Refinar Escenarios con Estados: Incorpora estados del sistema en los escenarios lógicos.
- Crear Modelo de Datos Conceptual ([CDB]): Identifica principales entidades de datos lógicas.
- Definir Cadenas Funcionales Lógicas ([LAB], [LFCD]): Muestra secuencias de funciones lógicas.

El refinamiento de funciones (paso 4) y la arquitectura lógica (paso 5) puede ser iterativo.

4 **Metodología para el Desarrollo de la Fase Física** Traduce la arquitectura lógica en una solución técnica concreta.

- Transición desde Fase Lógica: Adapta elementos lógicos a la perspectiva física.
- Elaborar Diagrama de Desglose de Componentes Físicos ([PCBD]): Define nodos de hardware y artefactos de software.

- Construir Diagrama de Arquitectura Física (Nodos) ([PAB]): Se centra en componentes hardware.
- Mapear Componentes de Comportamiento sobre Nodos ([PAB]): Asigna componentes lógicos (transicionados a físicos de comportamiento) a nodos hardware.
- Definir Conexionado Físico ([PAB]): Establece enlaces físicos entre componentes.
- Generar Diagrama de Arquitectura Física Completo ([PAB]): Integra todos los componentes físicos, sus intercambios (PEs, CEs) y el mapeo de funciones físicas (FEs).
- Mapear Intercambios entre Componentes Físicos: Detalla cómo los intercambios lógicos se materializan físicamente.
- Refinar Modelo de Datos ([CDB]): Detalla estructuras de datos físicas.
- Crear Diagramas de Interfaces Detallados ([CDI]): Especifica contratos de interfaces físicas.
- Mapear Interfaces sobre Puertos Físicos: Vincula interfaces a puertos de componentes físicos.
- Crear Cadenas Funcionales Físicas ([PAB], [PFCD]): Muestra secuencias de funciones implementadas físicamente.

2.4.2 Metodología MOFLT Airbus

La metodología MOFLT, desarrollada por Airbus Defence and Space (ADS), es un enfoque de Ingeniería de Sistemas Basada en Modelos (MBSE) que estructura el proceso de diseño y análisis de sistemas complejos. Su nombre es un acrónimo de las distintas capas o perspectivas que aborda: Misión, Operación, Función, Lógica y Técnica. A menudo se le añade una "R" inicial (R-MOFLT) para enfatizar la gestión de Requisitos (Requirements) a lo largo de todo el ciclo.

La inclusión explícita de la capa de Requisitos en la fase inicial del ciclo de vida subraya el reconocimiento de la gestión y trazabilidad de los requisitos como un pilar fundamental y un punto de partida esencial en la ingeniería de sistemas contemporánea. Esta práctica alinea el marco R-MOFLT de manera más estrecha con los procesos estandarizados de ingeniería, donde la elicitación, el análisis y la gestión de requisitos preceden y fundamentan el análisis de la misión y las fases subsecuentes de diseño. Si bien algunos documentos indican que la reutilización de requisitos, como concepto dentro del marco, se encontraba en etapas tempranas de maduración en el momento de su redacción, se enfatiza que la capa de requisitos constituye una consideración integral. La capacidad del marco para facilitar la trazabilidad entre los artefactos del modelo y los requisitos textuales es una manifestación de esta priorización. Esta evolución hacia R-MOFLT denota una transición hacia un enfoque más holístico y completo de la ingeniería de sistemas, enfatizando la necesidad crítica de anclar todo el proceso de desarrollo del sistema en requisitos claramente definidos, validados y rastreables a lo largo de todo el ciclo de vida.

A. **Capa de Requisitos (R - Requirement)** Esta capa se centra en el análisis, especificación y gestión de todos los requisitos que el sistema debe satisfacer. Aunque la madurez de las estrategias para la reutilización de requisitos ha sido un área de desarrollo continuo , la capa en sí misma es fundamental. Un aspecto crucial facilitado por esta capa, y por el marco en general, es la capacidad de establecer y mantener la trazabilidad entre los requisitos textuales (a menudo gestionados en herramientas dedicadas) y los diversos artefactos del modelo generados en las capas subsecuentes. Dicha trazabilidad es vital para la verificación, la validación, el análisis de impacto de cambios y la demostración de cumplimiento.

B. **Análisis de Misión (M - Mission Analysis)** La capa de Análisis de Misión (Mission Analysis) se adentra en el espacio del problema, buscando responder a la pregunta fundamental: ¿QUÉ problema necesita ser resuelto y CUÁLES son las formas potenciales de resolverlo?. Esta capa se ocupa de definir el propósito de alto nivel del sistema, sus objetivos principales, el concepto general de la misión, las capacidades que la misión debe proveer y las distintas fases por las que transcurrirá la misión. Debido a su naturaleza, el análisis de misión es inherentemente específico para cada empresa o proyecto particular. Por ejemplo, en el contexto de vehículos espaciales, esta capa se considera muy particular a la misión específica de la nave, lo que limita su genericidad y, por ende, su potencial para la reutilización directa de modelos entre plataformas con misiones dispares.

   - **Diagrama de Entorno:** Identifica todas las entidades externas que interactúan con el Sistema Industrial (como proveedores, clientes, autoridades, etc.) y los flujos de información y materiales entre ellos.
   - **Ciclo de Vida del Sistema:** Define las distintas fases por las que pasa el sistema, desde su concepción y desarrollo hasta su operación en serie, mantenimiento y eventual desmantelamiento.
   - **Formalización de Misiones:** Las necesidades de los stakeholders se traducen en "declaraciones de misión" estructuradas. Cada declaración debe especificar el contexto (fase del ciclo de vida), la contribución esperada del sistema y los objetivos de rendimiento cuantificables.

C. **Análisis Operacional (O - Operational Analysis)** Continuando dentro del espacio del problema, la capa de Análisis Operacional (Operational Analysis) define QUÉ hará el Sistema de Interés (SoI) para contribuir al cumplimiento de la misión y CUÁL es el contexto en el que operará dicho SoI. Esta capa gestiona la definición de las capacidades operacionales del sistema y las fases operacionales a través de las cuales se desplegarán estas capacidades. Esto incluye la identificación de los actores externos que interactuarán con el sistema, los escenarios operativos clave y los flujos de interacción. En términos de reutilización de modelos, el análisis operacional puede ofrecer un grado de genericidad a nivel de plataforma, pero esto suele estar condicionado a tipos específicos de sistemas o dominios de operación; por ejemplo, para naves espaciales, podría ser genérico para actividades en Órbita Baja Terrestre (LEO) o en Órbita Geoestacionaria (GEO), pero no necesariamente entre ambas sin adaptación.

   - **Escenarios Operacionales:** Se utilizan para ilustrar las interacciones cronológicas entre el sistema y los actores externos en diferentes contextos. Cada escenario se desencadena por un evento específico y cubre una o varias misiones.

D. **Arquitectura Funcional (F - Functional Architecture)** La capa de Arquitectura Funcional (Functional Architecture) marca la transición hacia el espacio de la solución. Su objetivo es responder a la pregunta: ¿CÓMO funcionará el Sistema de Interés para cumplir con las expectativas y los requisitos operacionales definidos previamente? Esta capa se enfoca en identificar y definir todas las funciones que el sistema debe realizar para proveer sus capacidades. Esto incluye la especificación de los intercambios funcionales (datos, energía, materia) entre estas funciones. Un principio clave en esta capa es la abstracción de la implementación física; las funciones se definen por lo que hacen, no por cómo se implementan físicamente. La arquitectura funcional a menudo se organiza jerárquicamente para gestionar la complejidad, descomponiendo funciones de alto nivel en subfunciones más detalladas hasta un nivel apropiado para la asignación a componentes lógicos. Esta capa es considerada particularmente apropiada para la reutilización de modelos, siempre que la descomposición funcional no se

lleve a un nivel de detalle excesivamente específico que la vincule prematuramente a una solución técnica particular.

- **Función:** Se define como un proceso que transforma unas entradas (materiales o información) en unas salidas para lograr un objetivo. Semánticamente, una función se nombra con un verbo de acción, como "Ensamblar componentes".
- **Arquitectura Funcional:** Es la combinación de dos perspectivas complementarias:
  - **Vista Estática (Desglose):** Descompone las funciones de alto nivel en sub-funciones más pequeñas, mostrando una estructura jerárquica.
  - **Vista Dinámica (Secuencia):** Muestra el orden en que se ejecutan las funciones y los flujos de información y materiales que las conectan, formando una cadena de procesos.

E. **Arquitectura Lógica (L - Logical Architecture)** Avanzando en el espacio de la solución, la capa de Arquitectura Lógica (Logical Architecture) describe CÓMO está organizado internamente el Sistema de Interés, centrándose en la definición de componentes abstractos o lógicos. Estos componentes lógicos son agrupaciones coherentes de funciones identificadas en la capa anterior. La arquitectura lógica define las responsabilidades de cada componente lógico, sus interfaces y las conexiones lógicas que soportan los intercambios funcionales. El objetivo es estructurar el sistema de una manera que sea conceptualmente clara para los arquitectos y que facilite la posterior asignación a elementos técnicos. Al igual que la arquitectura funcional, la arquitectura lógica se considera un buen candidato para la reutilización de modelos, especialmente si la definición de los componentes lógicos y sus interfaces se mantiene a un nivel de abstracción adecuado, sin descender a detalles de implementación específicos de hardware o software.

F. **Arquitectura Técnica (T - Technical Architecture)** La capa de Arquitectura Técnica (Technical Architecture) es la más concreta dentro del espacio de la solución y define CÓMO se implementará físicamente el Sistema de Interés. En esta capa se toman decisiones sobre los componentes de hardware y software específicos, las tecnologías a utilizar, y cómo los componentes lógicos se materializarán en elementos físicos y cómo estos se interconectarán. Debido a su naturaleza, la arquitectura técnica está intrínsecamente ligada a las elecciones tecnológicas y a los componentes físicos seleccionados para un proyecto específico. Esto hace que la reutilización directa de modelos de arquitectura técnica entre proyectos diferentes sea más desafiante. Si bien es posible proponer una vista técnica reutilizable basada en un catálogo de componentes, esto generalmente requiere un enfoque de ingeniería de línea de productos a nivel de empresa que se base en la reutilización de la arquitectura física. En muchos casos, el modelo del sistema a nivel técnico debe ser implementado o adaptado significativamente por el proyecto, basándose en las definiciones funcionales y lógicas que pueden haber sido extraídas de un modelo de plataforma genérico.

G. **Transición del Espacio del Problema al Espacio de la Solución** La estructura en capas de R-MOFLT facilita una transición sistemática y trazable desde la definición del espacio del problema (cubierto principalmente por las capas de Misión y Operacional) hacia la definición del espacio de la solución (desarrollado a través de las capas Funcional, Lógica y Técnica). Esta progresión estructurada es un principio central de la ingeniería de sistemas robusta. Asegura que las soluciones propuestas (arquitecturas F, L y T) se deriven lógicamente y se justifiquen a partir de una comprensión profunda del problema, los objetivos de la misión y el contexto operacional (capas M y O).

Esta organización en capas no es simplemente una forma de categorizar la información; actúa como un mecanismo fundamental para gestionar la complejidad inherente al diseño de

sistemas aeroespaciales. Cada capa se construye sobre la base de la anterior, creando una cadena de decisiones y artefactos que vinculan las necesidades de la misión de más alto nivel con los detalles de la implementación técnica. Esta estructura facilita la coherencia del diseño, ayuda a identificar posibles omisiones o inconsistencias y es crucial para realizar análisis de impacto efectivos cuando se producen cambios en los requisitos o en el entorno operativo.

La selectividad observada en la reutilización de las diferentes capas de R-MOFLT es una adaptación pragmática de la metodología que refleja una comprensión matizada de dónde reside el mayor valor de la reutilización en el ciclo de vida del desarrollo. Como se ha mencionado, las capas Funcional (F) y Lógica (L) son identificadas consistentemente como las más adecuadas para la construcción de modelos de plataforma genéricos y reutilizables. Esto se debe a que estas capas abstraen la funcionalidad esencial y la organización lógica del sistema de los detalles altamente específicos de una misión particular (capa M) o de las elecciones de componentes físicos y tecnológicos (capa T). La capa de Misión, por su naturaleza, está intrínsecamente ligada a los objetivos únicos de cada proyecto. Por otro lado, la capa Técnica, al estar tan próxima a la implementación física, tiende a variar significativamente entre proyectos a menos que exista una estrategia de línea de productos a nivel de empresa muy bien definida que estandarice componentes y arquitecturas físicas. Por lo tanto, al centrar los esfuerzos de reutilización en las capas F y L, se logra un "punto óptimo" que maximiza la aplicabilidad de los activos de modelado reutilizables a través de una gama más amplia de proyectos [European Space Agency, 2022].

2.4.3 La Metodología MagicGrid® de No Magic (Dassault Systèmes)

La metodología MagicGrid® es un enfoque de Ingeniería de Sistemas Basada en Modelos (MBSE) desarrollado por No Magic, actualmente parte de Dassault Systèmes [Dassault Systèmes, a]. Surge de la consolidación de buenas prácticas probadas en sectores de alta complejidad como el aeroespacial, la defensa y el automotriz. Aunque está estrechamente asociada con la herramienta de modelado Cameo Systems Modeler™, MagicGrid® es, por definición, una metodología agnóstica a la herramienta, compatible con cualquier entorno de modelado que soporte el estándar SysML sin requerir extensiones propietarias [Martínez-Val and Pérez, sf].

El marco metodológico se articula sobre una estructura matricial (la "parrilla" o *grid*) que organiza el proceso de desarrollo en dos dimensiones ortogonales, tal como se esquematiza en la **figura 2.4**:

Dominios de Abstracción (Filas de la Matriz): Definen la perspectiva y el nivel de detalle del sistema, alineándose conceptualmente con el ciclo de vida de la norma ISO/IEC/IEEE 15288.

- **Dominio del Problema (Caja Negra):** Se enfoca en el *qué*: las necesidades de los interesados, el comportamiento externo observable y las interacciones del sistema con su entorno, sin detallar la solución interna.
- **Dominio de la Solución (Caja Blanca):** Describe el *cómo*: la arquitectura interna del sistema, descomponiéndola en subsistemas y componentes, y asignando responsabilidades para satisfacer los requisitos.
- **Dominio de Implementación:** Este dominio, que abarcaría la realización física o la codificación de los componentes, **queda explícitamente fuera del alcance de la metodología MagicGrid®**, que se centra en las fases de especificación y diseño arquitectónico.

Pilares de Modelado (Columnas de la Matriz): Representan las cuatro vistas fundamentales del sistema según SysML.

- **Requisitos (Requirements):** Especificaciones formales de las capacidades que el sistema debe poseer.
- **Comportamiento (Behavior):** Dinámica del sistema, incluyendo funciones, estados e interacciones.
- **Estructura (Structure):** Organización de los constituyentes del sistema (subsistemas, componentes) y sus relaciones.
- **Paramétricos (Parametrics):** Modelado de restricciones cuantitativas y ecuaciones para análisis de ingeniería (rendimiento, masa, coste, etc.).

El proceso de modelado progresa a través de la matriz, generalmente de izquierda a derecha y de arriba hacia abajo, detallando progresivamente el sistema.

Dominio del Problema (Análisis de Caja Negra)

El desarrollo se inicia capturando las necesidades de alto nivel para definir el alcance y los criterios de éxito del sistema.

Pilar de Requisitos: Se capturan las necesidades de los interesados (*Stakeholder Needs*) y se refinan (`«refine»`) en Requisitos de Sistema formales.

Pilar de Comportamiento: Se modelan los Casos de Uso que describen la interacción del sistema con actores externos. Estos casos de uso se detallan mediante Diagramas de Actividad para definir las funciones principales del sistema.

Pilar de Estructura: Se define el Contexto del Sistema mediante un Diagrama de Bloques Interno (IBD), mostrando el sistema como un bloque único con sus puertos de interacción.

Pilar de Paramétricos: Se establecen las Medidas de Efectividad (MoE) que servirán como métricas clave para la validación del sistema.

Dominio de la Solución (Diseño de Caja Blanca): Nivel de Subsistemas

Una vez consolidada la visión externa, se diseña la arquitectura lógica del sistema.

Pilar de Requisitos: Los Requisitos de Sistema se descomponen y asignan, generando Requisitos de Subsistema más detallados mediante la relación de derivación (`«deriveReqt»`).

Pilar de Comportamiento: Las funciones de alto nivel se asignan (`«allocate»`) a los subsistemas responsables de ejecutarlas.

Pilar de Estructura: Se define la Arquitectura de Subsistemas mediante Diagramas de Bloques (BDD e IBD), descomponiendo el sistema en sus principales elementos lógicos y especificando sus interfaces.

Pilar de Paramétricos: Se refinan las MoE en Medidas de Rendimiento (MoP) más específicas, asociadas a los subsistemas.

Dominio de la Solución (Diseño de Caja Blanca): Nivel de Componentes

Se detalla la arquitectura hasta el nivel de los componentes que implementarán la solución.

Pilar de Requisitos: A su vez, los Requisitos de Subsistema se derivan en Requisitos de Componente, especificando las exigencias para los elementos de más bajo nivel de la arquitectura.

Pilar de Comportamiento: El comportamiento de cada componente se modela en detalle, a menudo con Diagramas de Máquina de Estados, para mostrar cómo reaccionan a eventos. Este comportamiento debe satisfacer (`«satisfy»`) los requisitos asignados.

Pilar de Estructura: Se detalla la estructura interna de cada componente. La arquitectura a este nivel también debe satisfacer («satisfy») los requisitos correspondientes.

Pilar de Paramétricos: Se desarrollan Diagramas Paramétricos detallados que vinculan las propiedades de los componentes con ecuaciones de análisis. Estos análisis validan cuantitativamente que se satisfacen («satisfy») los requisitos numéricos.

Trazabilidad Integral: A lo largo de todo el proceso, las relaciones semánticas de SysML («refine», «deriveReqt», «satisfy», «allocate») son cruciales. Crean una red de trazabilidad digital que conecta las necesidades originales de los interesados con los componentes específicos de la solución, garantizando que el diseño final sea completo, consistente y verificable.

2.5 Marco Teórico sobre Herramientas de modelado

En el presente capítulo se establece el marco teórico sobre las herramientas de software que posibilitan la aplicación de la metodología de Ingeniería de Sistemas Basada en Modelos (MBSE). Para ilustrar los distintos enfoques existentes en el mercado, se han seleccionado dos de las plataformas más representativas y paradigmáticas: Capella y Cameo Systems Modeler. Su análisis permite contrastar dos filosofías fundamentales en la aplicación de MBSE.

2.5.1 Capella: El Enfoque Centrado en el Método

Capella es una solución de modelado MBSE de código abierto, promovida por Thales, que se caracteriza por ser una herramienta *method-centric*. Su arquitectura de software está intrínsecamente acoplada a la metodología de ingeniería ARCADIA [Eclipse Foundation, 2025].

Este diseño implica que Capella ofrece un entorno de modelado altamente estructurado y prescriptivo. En lugar de proporcionar un conjunto de herramientas de dibujo genéricas, la plataforma guía activamente al ingeniero a través de un flujo de trabajo predefinido. La interacción del usuario con la herramienta está diseñada para hacer cumplir la coherencia y los principios del método ARCADIA, asegurando que el modelo resultante sea robusto y metodológicamente consistente. Dicha interfaz, que guía de manera muy clara los distintos pasos para el modelado en ARCADIA, se puede observar en la **Figura 2.5**.

Una de las diferencias más notables es su naturaleza *open-source*. Esta condición no solo implica la ausencia de costes de licencia, lo que democratiza su acceso para organizaciones de cualquier tamaño e instituciones académicas, sino que también fomenta una comunidad activa de usuarios y desarrolladores. Dicha comunidad contribuye a su evolución, a la creación de extensiones y a la disponibilidad de recursos compartidos. La plataforma se beneficia del robusto ecosistema de la Fundación Eclipse, que facilita la interoperabilidad y la extensibilidad mediante un sistema de *plugins* [Accenture and Navantia, sf].

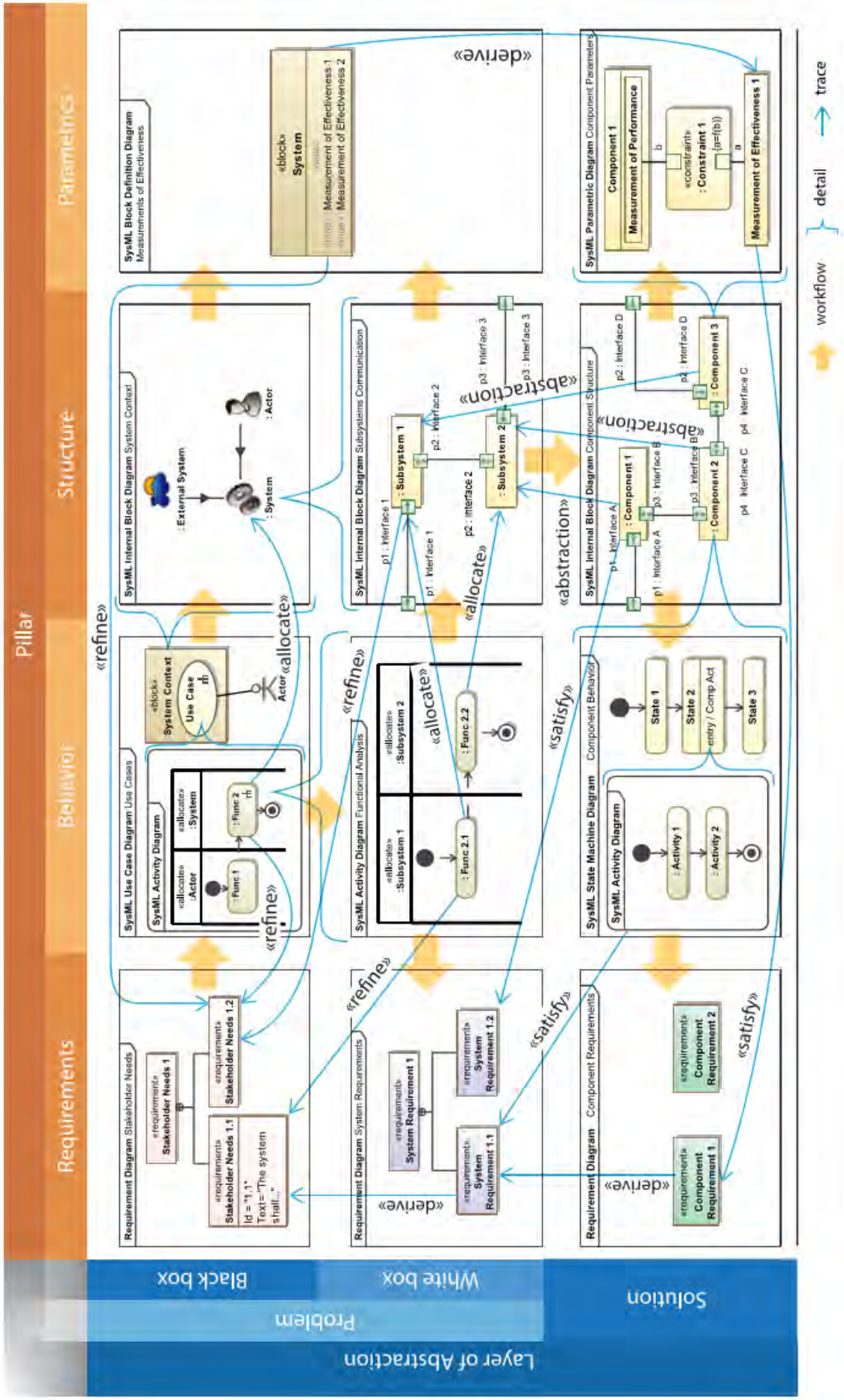

Figura 2.4 Visión general del proceso de modelado con la arquitectura MagicGrid.

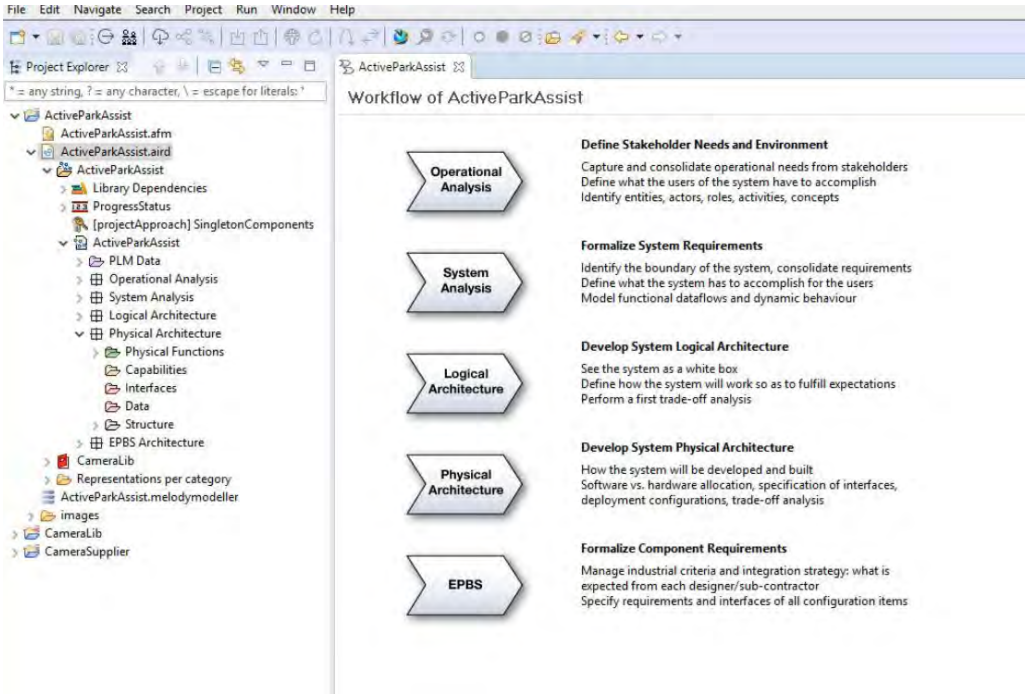

Figura 2.5 Interfaz de la herramienta Capella mostrando el desarrollo de la metodología ARCADIA.

Técnicamente, Capella utiliza un lenguaje de modelado específico del dominio (DSL), diseñado a medida para los conceptos de su metodología, en lugar del estándar SysML. El valor fundamental de Capella reside, por tanto, en su capacidad para estandarizar el proceso de diseño arquitectónico en una organización, garantizando un alto nivel de rigor, gobernanza y una trazabilidad completa como resultado directo de seguir su marco de trabajo. Es una solución optimizada para contextos donde la justificación del diseño y la madurez del proceso son críticas.

Frente a la flexibilidad que puede ofrecer una herramienta SysML genérica como Cameo, que permite una amplia libertad en la forma de modelar (siempre dentro del estándar), Capella puede percibirse como más prescriptiva. Sin embargo, esta prescriptividad es precisamente una de sus fortalezas para los equipos que buscan una guía clara y un marco de trabajo ya validado en la industria para la ingeniería de sistemas.

En cuanto a la extensibilidad, si bien las soluciones comerciales suelen ofrecer módulos adicionales y APIs, el carácter abierto de Capella permite un grado de personalización y desarrollo de extensiones (add-ons) que pueden ser específicos para un dominio industrial o una necesidad organizacional particular, sin depender necesariamente de la hoja de ruta de un proveedor comercial.

2.5.2 Cameo Systems Modeler: El Enfoque Centrado en el Lenguaje

En contraposición, Cameo Systems Modeler, de Dassault Systèmes, se define como una plataforma language-centric. Su base técnica es una implementación de alta fidelidad del estándar industrial OMG SysML® (Systems Modeling Language), un lenguaje de modelado de propósito general para la ingeniería de sistemas.

Este enfoque proporciona al equipo de ingeniería una "caja de herramientas" versátil y potente, basada en un lenguaje estandarizado y reconocido en la industria. A diferencia de Capella, Cameo no impone una metodología específica, lo que le confiere una gran flexibilidad para adaptarse a los procesos de ingeniería ya existentes en una organización o para implementar metodologías a medida.

Las fortalezas técnicas de Cameo radican en sus avanzadas capacidades analíticas. Permite no solo la descripción estructural, comportamental y de requisitos de un sistema, sino también su validación y verificación a través de:

- **Simulación de comportamiento:** La plataforma puede ejecutar los modelos de comportamiento para validar la lógica del sistema y detectar fallos de diseño de forma temprana.
- **Análisis cuantitativo:** Facilita la realización de análisis de trade-off y estudios de rendimiento mediante el modelado paramétrico, a menudo integrándose con solvers matemáticos externos (como MATLAB/Simulink) para resolver complejas ecuaciones de ingeniería.

Por tanto, el valor de Cameo reside en su flexibilidad, su estricta adherencia a los estándares industriales y sus potentes capacidades de simulación y análisis, que son fundamentales para la ingeniería de detalle y la verificación exhaustiva del diseño del sistema [Dassault Systèmes, 2025].

2.5.3 Síntesis Comparativa

En conclusión, la elección entre una herramienta como Capella o Cameo no representa una decisión sobre cuál es tecnológicamente superior, sino una decisión estratégica sobre el proceso de ingeniería que se desea implementar. Capella, con su enfoque guiado, es la elección preferente cuando se busca instaurar un proceso de diseño arquitectónico riguroso y estandarizado. Por otro lado, Cameo Systems Modeler es la opción adecuada cuando se requiere la flexibilidad de un lenguaje estándar como SysML y cuando las capacidades de análisis cuantitativo y simulación de comportamiento son críticas para la verificación y validación del diseño del sistema.

2.6 Reflexión sobre la Integración de Herramientas (MBSE, PLM, Simulación)

El desarrollo de arquitecturas de sistemas complejas, abordado desde la **Ingeniería de Sistemas Basada en Modelos (MBSE)**, resulta estratégicamente ineficaz si no se enmarca dentro de un ecosistema digital cohesionado. Dicho ecosistema debe estar gobernado por una plataforma de **Gestión del Ciclo de Vida del Producto (PLM)**.

La plataforma PLM se erige como el eje vertebrador de este ecosistema. Su función trasciende el mero almacenamiento de ficheros para convertirse en la gestora de las complejas interrelaciones entre los distintos artefactos de ingeniería. Mediante una robusta gestión de la configuración y el establecimiento de un **hilo digital (digital thread)** continuo, el PLM garantiza la coherencia y la trazabilidad de la información. La operación en silos de información, por contra, conduce a un modelo de trabajo propenso a errores, donde la trazabilidad entre un requisito, el componente físico que lo implementa y las pruebas que lo validan, se convierte en un proceso manual, insostenible y carente de fiabilidad en proyectos de alta complejidad.

Asimismo, el alcance del hilo digital debe extenderse a las fases posteriores al diseño, como son el mantenimiento y las operaciones, englobadas en la **Gestión del Ciclo de Vida del Servicio (SLM)**. La capacidad de rastrear un fallo detectado en servicio hasta su origen —ya sea un lote de producción, un análisis de simulación o una especificación de diseño— es crítica.

En conclusión, la integración descrita no es un complemento opcional, sino la estrategia fundamental que habilita la creación de un **Gemelo Digital Autoritativo**. Este no debe entenderse simplemente como un modelo 3D, sino como una red holística e interconectada de todos los datos, modelos y procesos que definen un producto a lo largo de su existencia. Dicha integración es la base sobre la que se construyen los principios de la Industria 4.0, permitiendo una toma de decisiones más rápida, informada y fiable en todas las etapas del ciclo de vida.

2.6.1 Suite de Siemens

Por Suite de Siemens se hace referencia al conjunto de productos de software y hardware desarrollados por Siemens que se utilizan en diversos campos, incluyendo la automatización industrial, el software PLM (Product Lifecycle Management), y la gestión de la cadena de suministro.

A partir del análisis de la industria y la información disponible, se puede afirmar que Siemens Xcelerator, junto con sus componentes fundamentales, se posiciona como una de las soluciones líderes y más extendidas en el ámbito del software industrial y la gestión del ciclo de vida del producto a nivel global [CIMdata, Inc., 2024]. Su presencia es particularmente significativa en sectores de alta complejidad y exigencia tecnológica. Se observa una fuerte penetración en la industria de automoción, donde tanto los grandes fabricantes (OEMs) como una parte considerable de su cadena de valor recurren a estas herramientas para el diseño, simulación y gestión de la producción. De manera similar, el sector aeroespacial y de defensa muestra una adopción elevada, impulsada por la necesidad de una gestión rigurosa de ciclos de vida extensos y productos tecnológicamente avanzados [Siemens Digital Industries Software, 2021].

Otras áreas donde la implantación de Siemens Xcelerator es notable incluyen la maquinaria industrial y equipos pesados, la electrónica y semiconductores (especialmente tras la integración de las soluciones de Mentor Graphics, ahora Siemens EDA), así como en el sector de energía y servicios públicos. Si bien es difícil cuantificar el número total de licencias o empresas, se estima que decenas de miles de organizaciones a nivel mundial, abarcando desde corporaciones multinacionales hasta pequeñas y medianas empresas (PYMES), utilizan componentes clave de esta suite.

La relevancia y la amplia difusión de Siemens Xcelerator pueden atribuirse a su enfoque integral del gemelo digital, la robustez de sus soluciones individuales para cada etapa del ciclo de vida (diseño, simulación, fabricación, operaciones) y la capacidad de integración que ofrece su plataforma Teamcenter. La tendencia hacia modelos de "Software as a Service" (SaaS) con Xcelerator as a Service también está facilitando su acceso y adopción por un espectro más amplio de empresas, consolidando su posición como un actor fundamental en la digitalización de la industria [Siemens Digital Industries Software, 2024].

A continuación se muestran algunos de los softwares que podrían ser de interés si se utiliza esta Suite para modelado y simulación.

NX

NX (anteriormente conocido como Unigraphics) es una de las soluciones de software más avanzadas y completas para el **Diseño Asistido por Ordenador (CAD)**, la **Ingeniería Asistida por Ordenador (CAE)** y la **Fabricación Asistida por Ordenador (CAM)**. Está diseñado para gestionar la totalidad del proceso de desarrollo de productos en un entorno integrado y de alto rendimiento.
Características y Roles Clave:

- **Diseño y Modelado (CAD):** Ofrece potentes herramientas para el modelado de sólidos, superficies avanzadas (esencial en aerodinámica), chapa metálica, y diseño de grandes ensamblajes. Soporta tanto modelado paramétrico como síncrono (una combinación de modelado directo y con historial), lo que proporciona una flexibilidad excepcional en el diseño y la modificación de geometrías complejas.
- **Simulación y Análisis (CAE):** Integra capacidades de simulación robustas, permitiendo a los ingenieros realizar análisis de elementos finitos (FEA) para estudios estructurales, térmicos, de flujo y de movimiento directamente sobre el modelo de diseño. Esta integración agiliza la validación y optimización del producto sin necesidad de exportar la geometría a herramientas externas.
- **Fabricación (CAM):** Proporciona una solución completa para la programación de máquinas de control numérico (CNC), desde fresado de 2.5 ejes hasta mecanizado complejo de 5 ejes

simultáneos, así como torneado y programación de robots. Permite generar trayectorias de herramienta optimizadas directamente a partir del modelo 3D.

- **Ingeniería Electromecánica:** Facilita el diseño colaborativo de sistemas que integran componentes mecánicos, eléctricos y electrónicos, siendo una pieza clave para el desarrollo de productos mecatrónicos.
- **Gemelo Digital:** Es una herramienta fundamental para la creación del gemelo digital de un producto, ya que contiene la definición geométrica y funcional precisa que se utilizará en las fases posteriores de simulación y fabricación.

Teamcenter

Teamcenter es la plataforma de **Gestión del Ciclo de Vida del Producto (PLM)** de Siemens. Actúa como la columna vertebral digital que conecta personas, procesos y conocimiento a lo largo de toda la vida del producto. Su función principal es gestionar toda la información y los procesos asociados a un producto, desde la concepción y el diseño hasta la fabricación, el servicio y el fin de su vida útil. En la **Figura 2.6** se muestra bien el papel central de Teamcenter como plataforma central por la que pasa la información de todas las distintas herramientas como SMW (System Modeling Workbench), Amesim y HEEDS MDO.

Características y Roles Clave:

- **Fuente Única de Verdad (Single Source of Truth):** Centraliza y gestiona de forma segura todos los datos relacionados con el producto (modelos CAD de NX y otros sistemas, documentos, especificaciones, software, etc.), asegurando que todos los implicados trabajen con la información más reciente y correcta.
- **Gestión de la Lista de Materiales (BOM):** Permite la creación y gestión de listas de materiales complejas y multidisciplinares (mecánicas, eléctricas, de software). Facilita la gestión de variantes y configuraciones de producto, algo crítico en industrias como la aeroespacial y la automotriz.
- **Automatización de Procesos y Flujos de Trabajo (Workflow):** Define, automatiza y gestiona los procesos de negocio, como las revisiones de diseño, las órdenes de cambio de ingeniería (ECO) y los flujos de aprobación. Esto garantiza la trazabilidad, el cumplimiento de normativas y la eficiencia operativa.
- **Colaboración Global:** Ofrece herramientas para que equipos distribuidos geográficamente puedan colaborar de manera efectiva, compartiendo y visualizando datos de forma segura.
- **Plataforma de Integración:** Es el núcleo que integra las diferentes herramientas de la suite de Siemens (como NX, Simcenter, y software de EDA) y también se conecta con otros sistemas empresariales como los ERP (Enterprise Resource Planning), creando un ecosistema digital cohesionado.
- **Gestión de Requisitos:** Permite capturar, gestionar y trazar los requisitos del cliente y los requisitos técnicos a lo largo de todo el ciclo de vida, asegurando que el producto final cumpla con todas las especificaciones.

Simcenter Amesim (Parte de Simcenter)

Simcenter Amesim es una plataforma líder para la simulación de sistemas mecatrónicos en 1D (unidimensional). Permite a los ingenieros modelar, simular y analizar el comportamiento de sistemas complejos que involucran múltiples dominios físicos antes de la construcción de prototipos.

Características y Roles Clave:

- **Multi-dominio:** Modela y simula la interacción entre componentes de diversos dominios físicos (fluidos, mecánica, eléctrica, control, termodinámica) en un solo entorno.

- **Bibliotecas Validadas:** Ofrece extensas bibliotecas de componentes predefinidos y validados para agilizar el modelado.
- **Análisis de Rendimiento:** Facilita la evaluación del rendimiento dinámico de los sistemas en condiciones de operación variadas.
- **Validación Temprana (Frontloading):** Ayuda a identificar y resolver problemas de diseño en las etapas iniciales del ciclo de desarrollo, reduciendo costes y tiempo.
- **Integración:** Se conecta de manera fluida con otras herramientas de Simcenter, software de CAD y la plataforma de gestión del ciclo de vida del producto (PLM) Teamcenter.
- **Aplicaciones:** Es ampliamente utilizado en industrias de alta complejidad como la aeroespacial, automotriz, maquinaria industrial y energía.

Simcenter Amesim (Parte de Simcenter)

Simcenter Amesim es una plataforma líder para la simulación de sistemas mecatrónicos en 1D (unidimensional). Permite a los ingenieros modelar, simular y analizar el comportamiento de sistemas complejos que involucran múltiples dominios físicos antes de la construcción de prototipos.

Características y Roles Clave:

- **Multi-dominio:** Modela y simula la interacción entre componentes de diversos dominios físicos (fluidos, mecánica, eléctrica, control, termodinámica) en un solo entorno.
- **Bibliotecas Validadas:** Ofrece extensas bibliotecas de componentes predefinidos y validados para agilizar el modelado.
- **Análisis de Rendimiento:** Facilita la evaluación del rendimiento dinámico de los sistemas en condiciones de operación variadas.
- **Validación Temprana (Frontloading):** Ayuda a identificar y resolver problemas de diseño en las etapas iniciales del ciclo de desarrollo, reduciendo costes y tiempo.
- **Integración:** Se conecta de manera fluida con otras herramientas de Simcenter, software de CAD y la plataforma de gestión del ciclo de vida del producto (PLM) Teamcenter.
- **Aplicaciones:** Es ampliamente utilizado en industrias de alta complejidad como la aeroespacial, automotriz, maquinaria industrial y energía.

Systems Modeling Workbench (SMW) para Teamcenter

Systems Modeling Workbench (SMW) es una solución de Siemens que se integra directamente en Teamcenter para dar soporte a la Ingeniería de Sistemas Basada en Modelos (MBSE). Es la misma aplicación de Capella aunque con el Add-on de Teamcenter para su inserción en el PLM.

Características y Roles Clave:

- **Definición de Arquitectura del Sistema:** Permite definir, analizar y gestionar la arquitectura completa del producto de manera formal.
- **Modelado Jerárquico:** Facilita la creación de modelos de sistema jerárquicos, descomponiendo la complejidad en partes manejables.
- **Trazabilidad:** Garantiza la trazabilidad digital completa entre los requisitos del sistema, la arquitectura lógica, el diseño físico, el análisis y las pruebas.
- **Colaboración:** Permite que equipos de ingeniería de diferentes disciplinas colaboren en un modelo de sistema único y centralizado.
- **Integración con Herramientas de Análisis:** Se conecta con herramientas de simulación (como Amesim) para validar el comportamiento y el rendimiento del sistema.

- **Reutilización de Modelos:** Promueve la reutilización de modelos y componentes de sistemas en diferentes proyectos y variantes de productos.

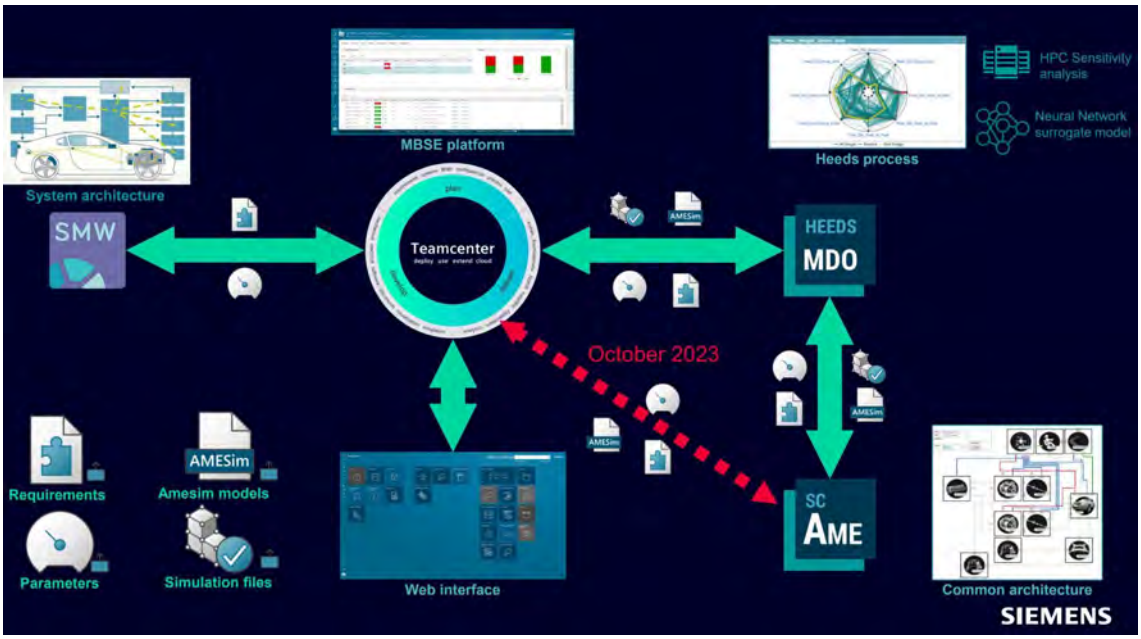

Figura 2.6 Diagrama del ecosistema de Siemens para MBSE.

2.6.2 Plataforma 3DEXPERIENCE de Dassault Systèmes

La plataforma **3DEXPERIENCE** de Dassault Systèmes es un entorno de negocio e innovación que proporciona a las organizaciones una visión holística y en tiempo real de su ecosistema empresarial. Funciona como una plataforma unificada que conecta personas, ideas, datos y soluciones en un único espacio colaborativo, cubriendo todo el espectro de actividades de la empresa, desde la ingeniería y la fabricación hasta el marketing y las ventas.

A partir del análisis de la industria, se puede afirmar que la plataforma 3DEXPERIENCE se posiciona como una de las soluciones líderes a nivel mundial en la Gestión del Ciclo de Vida del Producto (PLM) y la transformación digital [CIMdata, Inc., 2023]. Su presencia es dominante en sectores de alta tecnología donde la complejidad del producto es un factor crítico. El sector **aeroespacial y de defensa** ha sido históricamente uno de sus principales baluartes, con gigantes como Boeing y Airbus utilizándola extensivamente para el diseño, simulación y fabricación de sus aeronaves [Dassault Systèmes, 2017]. De manera similar, la industria de la **automoción y el transporte** depende en gran medida de esta plataforma para innovar, con empresas como Tesla y Renault que la emplean para gestionar todo el ciclo de vida de sus vehículos [Dassault Systèmes, 2021].

Otras áreas donde la implantación de la plataforma 3DEXPERIENCE es muy significativa incluyen la maquinaria industrial, la alta tecnología, la arquitectura e ingeniería, y las ciencias de la vida. Se estima que más de 300,000 empresas de todos los tamaños, desde corporaciones multinacionales hasta startups, utilizan las soluciones de Dassault Systèmes a nivel mundial.

La relevancia de la plataforma 3DEXPERIENCE se atribuye a su enfoque integral del **gemelo virtual (Virtual Twin)**, que permite crear una réplica digital viva no solo del producto, sino también de los procesos de producción y las operaciones. Su arquitectura nativamente integrada (a diferencia de una suite de productos adquiridos) y su disponibilidad en la nube (**3DEXPERIENCE on Cloud**) facilitan la colaboración y el acceso a herramientas de alto rendimiento, consolidando su posición como un pilar de la digitalización industrial [Dassault Systèmes, b].

A continuación se muestran algunas de las aplicaciones clave que nos podrían interesar si se utiliza esta Plataforma para modelado y simulación.

CATIA

CATIA (Computer-Aided Three-dimensional Interactive Application) es la aplicación de Dassault Systèmes para el **Diseño de Producto y la Ingeniería Asistida por Ordenador (CAD/CAE)**. Dentro de la plataforma 3DEXPERIENCE, CATIA va más allá del CAD tradicional para ofrecer una solución de ingeniería social y multidisciplinar, desde el concepto inicial hasta la experiencia final del producto.

Características y Roles Clave:

- **Diseño y Modelado (CAD):** Ofrece herramientas de vanguardia para el modelado de sólidos, superficies de Clase-A (esencial para aerodinámica y diseño de automoción), diseño generativo, modelado de compuestos y gestión de ensamblajes a gran escala. Su entorno de modelado híbrido proporciona una flexibilidad inigualable.
- **Ingeniería de Sistemas Basada en Modelos (MBSE):** A través de la integración de **CATIA Magic** (anteriormente No Magic), permite definir, analizar y gestionar arquitecturas de sistemas complejas utilizando el estándar SysML. Conecta los requisitos funcionales, lógicos y físicos (FLP) en un modelo de sistema unificado.
- **Simulación y Análisis (CAE):** Integra capacidades de análisis de elementos finitos (FEA) directamente en el entorno de diseño, permitiendo a los ingenieros validar el rendimiento estructural, térmico y cinemático sin abandonar la interfaz de CATIA, agilizando el proceso de diseño-validación.
- **Ingeniería de Fluidos y Eléctrica:** Facilita el diseño y enrutamiento 3D de sistemas de tuberías, HVAC y arneses eléctricos, garantizando la integración y coherencia espacial dentro del gemelo virtual.
- **Gemelo Virtual:** Es la aplicación fundamental para la creación del gemelo virtual de un producto, definiendo con la máxima fidelidad no solo la geometría, sino también su comportamiento, tolerancias funcionales (FT&A) y la definición de materiales.

ENOVIA

ENOVIA es la aplicación de **Gestión del Ciclo de Vida del Producto (PLM)** de la plataforma 3DEXPERIENCE. Actúa como la columna vertebral para la gobernanza de datos y la colaboración empresarial, conectando a todas las partes interesadas con una fuente única de verdad a lo largo de todo el ciclo de vida del producto.

Características y Roles Clave:

- **Fuente Única de Verdad (Single Source of Truth):** Centraliza y gestiona de forma segura todos los datos relacionados con el producto (modelos CAD de CATIA y otros sistemas, documentos, software embebido, etc.), asegurando la integridad y la trazabilidad de la información.
- **Gestión de la Lista de Materiales (BOM):** Permite la creación y gestión de listas de materiales multidisciplinarias y multinivel (eBOM, mBOM, sBOM). Es fundamental para la gestión de la variabilidad y la configuración de productos complejos.
- **Automatización de Procesos y Flujos de Trabajo:** Gestiona procesos de negocio clave como la gestión de cambios de ingeniería (ECO), la calidad (QMS) y el cumplimiento normativo mediante flujos de trabajo automatizados que garantizan la trazabilidad y la eficiencia.
- **Colaboración Global y en la Cadena de Suministro:** Ofrece herramientas para que equipos distribuidos geográficamente y socios externos puedan colaborar de manera segura y en tiempo real sobre los datos del producto.

- **Plataforma de Gobernanza:** Integra todas las aplicaciones de la plataforma 3DEXPERIENCE, asegurando que los datos generados en CATIA, SIMULIA o DELMIA estén gestionados y sean consistentes a lo largo del ciclo de vida.
- **Gestión de Requisitos:** Permite capturar, gestionar y trazar los requisitos del producto a lo largo del proceso de desarrollo, asegurando que el diseño final cumpla con todas las especificaciones.

SIMULIA y Dymola (Parte de SIMULIA)

SIMULIA es la marca de Dassault Systèmes para la **simulación realista multifísica y multiescala**. Se basa en un potente portafolio de tecnologías, incluyendo el solver Abaqus para FEA. **Dymola**, integrada dentro de SIMULIA, es la herramienta líder para la simulación de sistemas en 1D, basada en el lenguaje abierto Modelica.

Características y Roles Clave (Dymola/Sistemas 1D):

- **Multi-dominio:** Modela y simula la interacción dinámica entre componentes de múltiples dominios físicos (mecánica, eléctrica, control, térmica, hidráulica, neumática) en un único entorno. Ideal para sistemas mecatrónicos complejos.
- **Basado en el Lenguaje Modelica:** Utiliza el estándar abierto Modelica, lo que permite la creación de modelos reutilizables y la interoperabilidad. Dispone de extensas bibliotecas comerciales para automoción, aeroespacial, energía, etc.
- **Análisis de Rendimiento del Sistema:** Permite evaluar el rendimiento dinámico de sistemas completos (ej., un tren de potencia, un sistema de control de vuelo) bajo diversas condiciones de operación.
- **Validación Temprana (Frontloading):** Facilita la validación de la arquitectura y el rendimiento del sistema en las fases conceptuales del diseño, reduciendo drásticamente la necesidad de prototipos físicos.
- **Integración:** Se conecta de forma nativa con los modelos 3D de CATIA para la simulación 1D-3D (co-simulación) y con ENOVIA para la gestión de modelos y resultados, asegurando la consistencia dentro de la plataforma.
- **Aplicaciones:** Es una herramienta de referencia en la industria aeroespacial para el diseño y validación de sistemas de control ambiental (ECS), trenes de aterrizaje y sistemas de control de vuelo.

Cameo (para MBSE)

Cameo es la solución integrada en la plataforma 3DEXPERIENCE para la **Ingeniería de Sistemas Basada en Modelos (MBSE)**. Es la implementación del portafolio de No Magic (líder en el mercado de MBSE) dentro del ecosistema de Dassault Systèmes.

Características y Roles Clave:

- **Definición de Arquitectura del Sistema:** Permite a los ingenieros de sistemas definir, analizar y visualizar la arquitectura completa del producto utilizando lenguajes estándar como SysML, UML y UPDM.
- **Modelado Jerárquico:** Facilita la descomposición de sistemas complejos en subsistemas y componentes manejables, desde la definición de la misión hasta los componentes físicos individuales.
- **Trazabilidad Digital Continua:** Garantiza una trazabilidad completa y bidireccional entre los requisitos, el análisis de seguridad, la arquitectura del sistema, el diseño 3D (CATIA), la simulación 1D (Dymola) y las pruebas.
- **Colaboración Multidisciplinar:** Permite que ingenieros de software, hardware, seguridad y otras disciplinas colaboren sobre un único modelo de sistema compartido y gestionado en ENOVIA.

- **Ejecución y Verificación de Modelos:** Permite ejecutar los modelos de comportamiento del sistema para verificar que los requisitos se cumplen antes de escribir una sola línea de código o fabricar una pieza.
- **Reutilización de Arquitecturas:** Promueve la reutilización de modelos de sistema y patrones de arquitectura a través de diferentes líneas de productos, acelerando la innovación.

Comparativa de Enfoques: 3DEXPERIENCE vs. Siemens Xcelerator

Si bien ambas plataformas buscan ofrecer una solución integral para el gemelo digital, sus filosofías y arquitecturas fundamentales presentan diferencias clave, especialmente relevantes desde una perspectiva de ingeniería.

- **Filosofía Central:**
  - **Dassault Systèmes (3DEXPERIENCE):** Su enfoque es el de una **plataforma de negocio unificada**. No se presenta como una "suite" de productos, sino como un único entorno operativo donde las aplicaciones (CATIA, ENOVIA, SIMULIA) son nativas. La estrategia es “todo en uno”, buscando una integración de datos y procesos sin fisuras bajo un modelo de datos común. Es una visión más *top-down* y holística.
  - **Siemens (Xcelerator):** Su enfoque se centra en un **portfolio completo e integrado**. La estrategia se basa en la adquisición de las mejores herramientas en su clase (NX, Mentor, Simcenter) y su integración a través de la robusta plataforma PLM Teamcenter. La filosofía es más *bottom-up*, ofreciendo una mayor flexibilidad y apertura para integrar también herramientas de terceros.
- **Arquitectura y Origen:**
  - **DS:** La plataforma ha evolucionado orgánicamente desde CATIA y ENOVIA. Esto resulta en una **integración nativa** muy profunda entre sus componentes principales, ya que fueron diseñados desde el principio para trabajar juntos. El dato es el centro, y las apps giran a su alrededor.
  - **Siemens:** La suite es el resultado de la **integración de adquisiciones estratégicas**. Esto le confiere una enorme fortaleza en dominios específicos, como el diseño electromecánico (EDA con Mentor Graphics) y la simulación mecatrónica. Teamcenter actúa como el potente pegamento que cohesiona este diverso y potente ecosistema.
- **Fortalezas Clave en Aplicación:**
  - **DS:** Tradicionalmente insuperable en **diseño de superficies complejas (styling) y composites** (CATIA). Actualmente, es el líder indiscutible en **Ingeniería de Sistemas Basada en Modelos (MBSE)** gracias a la integración de CATIA Magic, un aspecto crítico para la definición de arquitecturas en sistemas aeroespaciales modernos.
  - **Siemens:** Muy fuerte en **diseño electromecánico y mecatrónica**, unificando el mundo mecánico (MCAD con NX) y electrónico (ECAD/EDA). Su herramienta de modelado síncrono en NX ofrece una flexibilidad excepcional para la edición de geometrías. La suite Simcenter es extremadamente completa para la simulación 1D y 3D.
- **Ecosistema y Flexibilidad:**
  - **DS:** Promueve un ecosistema más cerrado para garantizar la integridad de la plataforma. La experiencia es más homogénea y consistente si se adoptan todas sus soluciones, pero puede ser más rígida a la hora de integrar software externo.
  - **Siemens:** Promueve un ecosistema más **abierto y flexible**. Su arquitectura está diseñada para facilitar la integración con otros sistemas (ERP, y también software de la competencia), lo que puede ser una ventaja para empresas con un panorama de TI heterogéneo.

En resumen, para un ingeniero aeroespacial, la elección entre ambas plataformas podría simplificarse (con matices) de la siguiente manera: **3DEXPERIENCE** ofrece una visión unificada y profundamente integrada desde la arquitectura de sistemas (MBSE) hasta el diseño de composites, ideal para programas nuevos que buscan una única fuente de verdad. **Siemens Xcelerator** ofrece un portfolio de herramientas potentísimo y más abierto, ideal para la optimización de sistemas electromecánicos complejos y para entornos que valoran la flexibilidad y la integración con sistemas existentes.

2.6.3 Validación y Verificación mediante la simulación de la arquitectura

El interés del proyecto es la conexión entre modelado de arquitectura y simulación, que es lo que se ha hecho en este trabajo de fin de grado aunque a una escala menor. La validación temprana de la arquitectura del sistema es un paso crítico en el ciclo de vida del desarrollo, permitiendo identificar y corregir posibles deficiencias antes de incurrir en costes elevados de implementación y posteriores modificaciones. Se propone y aplica un proceso iterativo de validación de la arquitectura, y potenciado por la simulación. Este proceso se esquematiza en el diagrama de flujo presentado en la Figura 2.7, y se detalla a continuación.

Como ya se ha señalado más veces, el punto de partida de este proceso radica en la fase de Ingeniería de Sistemas, que abarca desde la definición de necesidades hasta el modelado a nivel físico. Durante estas etapas, se elaboran modelos progresivamente más detallados (Operacional, Sistema, Lógico y Físico), asegurando en todo momento la trazabilidad de los requisitos (PRBS, FBB) con los componentes del sistema y subsistemas. Esta trazabilidad es fundamental para garantizar que el diseño arquitectónico final se alinee directamente con las necesidades y restricciones iniciales.

Una vez que la arquitectura ha sido definida y parametrizada, se procede a la publicación de la arquitectura en una plataforma PLM, un paso formal que consolida el diseño hasta ese momento. Es en este punto donde se introduce una primera decisión crítica: ¿Se requiere un cambio en la arquitectura? Esta pregunta surge de revisiones internas, análisis de coherencia o la aparición de nuevas restricciones/oportunidades. Si la respuesta es afirmativa, el proceso retrocede para definir el alcance de requisitos de Validación/Sistema (VRs/SRs)[1], lo que implica una revisión de las necesidades y una reentrada en las fases de modelado si es necesario, estableciendo así un bucle de retroalimentación iterativo en la fase de diseño.

Si la arquitectura se considera estable y no requiere cambios inmediatos, se procede a enviar la Información de Solicitud de Validación (VRI) a un encargado, que servirá de enlace con la fase de Simulación.

La fase de Simulación constituye el núcleo de la validación experimental de la arquitectura. En esta etapa:

- **Creación del Modelo de Simulación:** Basándose en la arquitectura parametrizada, se desarrolla un Modelo de Simulación. Este modelo replica el comportamiento del sistema o de sus componentes clave.
- **Ejecución del Modelo de Simulación:** El modelo es ejecutado bajo diversas condiciones y escenarios, diseñados para poner a prueba la arquitectura frente a los requisitos establecidos y los comportamientos esperados.
- **Evaluación de Resultados:** Los datos obtenidos de la simulación son minuciosamente analizados. El objetivo es determinar si el rendimiento y el comportamiento del sistema simulado cumplen con los Requisitos de Validación/Sistema (VRs/SRs).

[1] **SR (Requisitos del Sistema)**: Son las especificaciones detalladas de lo que el sistema debe hacer y cómo. Sirven para verificar si el sistema se está construiendo correctamente según el diseño.

VR (Requisitos de Validación): Son los criterios para confirmar que el sistema final (o su modelo) cumple con su propósito real y las necesidades del usuario. Sirven para validar si se está construiguiendo el sistema adecuado para el problema.

Los resultados de esta evaluación conducen a una segunda decisión crucial: ¿Se aprueban los Requisitos de Validación/Sistema?

- Si la simulación demuestra que los requisitos son **Aprobados**, indica que la arquitectura se comporta según lo previsto y satisface sus objetivos. El proceso avanza hacia la Revisión de Resultados.
- Si los requisitos **Falla** en la simulación, significa que la arquitectura presenta deficiencias o no cumple con las expectativas. En este caso, el proceso también pasa a la Revisión de Resultados, pero con la implicación de que se han identificado problemas.

La etapa de revisar resultados es vital. Independientemente de si la simulación fue aprobada o fallida, esta revisión permite comprender en profundidad las implicaciones de los resultados. Si se identifican desviaciones significativas, oportunidades de mejora o la necesidad de ajustar la arquitectura (especialmente en caso de fallo), la información obtenida retroalimenta el proceso. Esta retroalimentación puede implicar una nueva Definición del Alcance de VRs/SRs, lo que a su vez puede conducir a una revisión de la arquitectura en la fase de Ingeniería de Sistemas, cerrando un bucle iterativo que busca la optimización continua, tal como se ilustra en la Figura 2.7.

Este enfoque garantiza que el desarrollo de diseño de un sistema sea un proceso de aprendizaje continuo, donde la simulación actúa como una herramienta poderosa para mitigar riesgos y asegurar la robustez del diseño arquitectónico.

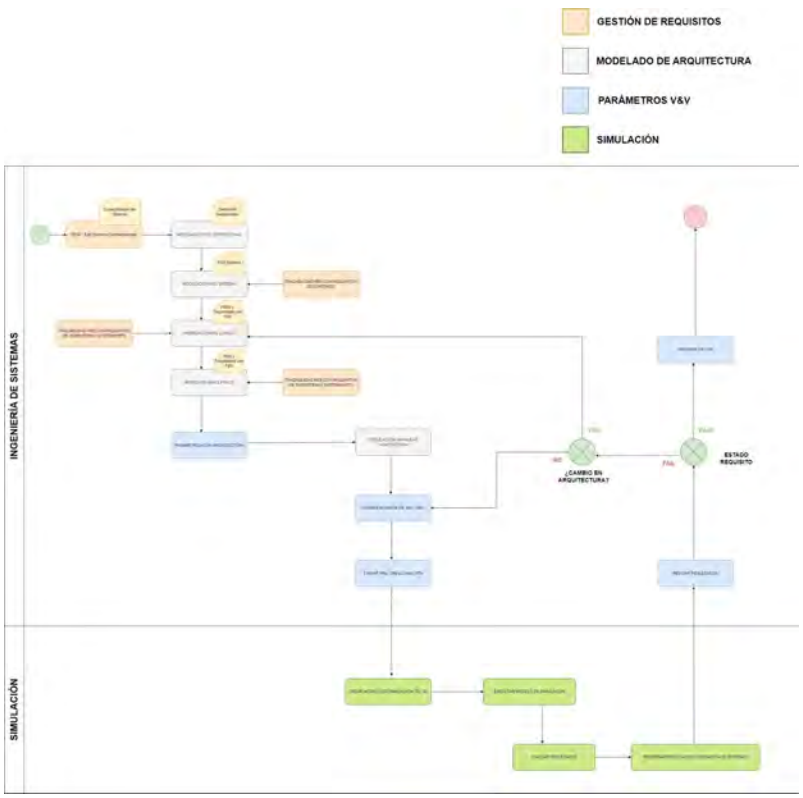

Figura 2.7 Flujograma de un proceso de trabajo basado en MBSE que integra el modelado de la arquitectura de sistema con la simulación para la Verificación y Validación (V&V).

3 Ejercicio Práctico. Sistema de Aire Acondicionado y Presurización de Cabina

3.1 Metodología

A continuación se expondrá la metodología de aplicación del **MBSE** para el diseño de un sistema de presurización y aire acondicionado de un avión basado en el del **C-295**. Se ha elegido este avión porque se ha encontrado información considerable para su dimensionamiento y proyecto, en especial ha sido útil un TFM [Peral González, 2018].

El problema es que habrá lagunas en muchos puntos, ya que no se manejan los documentos técnicos de Airbus que explican cómo funcionan los distintos compresores, turbinas,... o cuestiones del tipo: a qué temperatura se extrae el aire de los motores. Ni tampoco se tendrán requisitos impuestos por parte de Airbus. Estos requisitos normalmente son producto de años y años de experiencia de la empresa en el desarrollo de los distintos sistemas y solo son accesibles por los trabajadores de la compañía o de las subcontratas que trabajan con esos subsistemas.

Por lo tanto, ante la ausencia de datos específicos de propietario, se recurrirá a soluciones lógicas y justificadas, basadas en sistemas análogos y en la normativa pública. Cuando los requisitos específicos no sean accesibles, se formularán hipótesis realistas y fundamentadas para permitir el desarrollo del modelo. Al final, lo importante es poner sobre la mesa la tecnología, aunque los inputs no sean del todo correctos. Donde no se pueda expresar exactamente cómo es algún componente del sistema de aire acondicionado del C-295, se hará uso de aviones similares para disponer de un componente realista de cómo podría ser, sin ser exactamente el mismo de la aeronave de estudio.

Una de las características de la metodología **ARCADIA** es su razonamiento *top-to-bottom*. Por lo tanto, se partirá de lo más general, esto es, lo que se necesita del sistema, lo que se le solicita que haga, y se irán desmembrando capacidades operativas en funciones, subfunciones y actividades, hasta llegar al componente más simple.

En este trabajo se irán desarrollando los puntos desde lo más general hasta lo específico, respondiendo las preguntas de cada capa, poco a poco y en cada sección, analizando e investigando cómo funciona el sistema ACPC a la escala que se solicita. Una vez investigado y obtenido la información necesaria para realizar el modelado, se procederá con su desarrollo.

Análisis Operacional

Se tratará al sistema en su conjunto, como un todo homogéneo, sin partes. Y se identificarán cuáles son sus *stakeholders*: las personas que solicitarán a nuestro sistema que les solucione un problema.

A estas funciones generales que hará nuestro sistema, se les llamará **capacidades**. Estas capacidades dependerán únicamente de los *stakeholders* y no señalan cómo lo hará el sistema para cumplirlas.

Análisis de Sistema

En esta parte se buscará ver con qué sistemas trabaja conjuntamente nuestro sistema para ofrecer la solución. Estos lugares en los que se desarrollan estas interacciones serán los **escenarios**. Aquí se sigue tratando al sistema como uno solo. Se analizará cómo lo tiene que hacer el sistema para conseguir proveer estas capacidades.

Análisis Lógico

En esta fase, el objetivo es definir la **arquitectura lógica** del sistema. Ya no se ve al sistema como una caja negra monolítica, sino que se empieza a descomponer en componentes lógicos abstractos. Estos componentes lógicos no son todavía piezas físicas de hardware o software específicas, sino agrupaciones funcionales que colaboran para realizar las funciones del sistema identificadas en la fase anterior.

Análisis Físico

En esta fase, el objetivo es definir la **arquitectura física** del sistema. Se seleccionan y especifican los componentes de hardware y software reales que realizarán las funciones asignadas a los componentes lógicos. También se definen las interconexiones físicas entre ellos.

Conforme se vaya modelando la arquitectura de nuestro sistema se deben ir enlazando los requisitos. Simplemente se hará uso de los **requisitos de diseño** establecidos por la European Union Aviation Safety Agency (EASA), ya que no se tendrán requisitos de misión, ni de clientes, ni de cualquier otro tipo accesible para nuestro modelo; además, para demostrar la gestión de requisitos por parte de Capella tampoco es necesario profundizar en exceso.

Habrá requisitos que hablen de la propia arquitectura, de las redundancias mínimas que deben tener los componentes del sistema, de su fiabilidad y de las condiciones del aire que se tendrá en la cabina para las distintas fases de vuelo. Para ello, se deberán integrar diversos modelos de simulación en Capella llamándolos con **Python4Capella**. Se irán definiendo valores de dimensionamiento a la entrada del modelo de simulación y, según los resultados de la simulación, se verá si quedan satisfechos ciertos requisitos.

En concreto se realizarán dos tipos de simulaciones. Una será un modelo simplificado de un **ECS (Environmental Control System)** en Simulink y otra un modelo programado en MATLAB de un **análisis FTA (Fault Tree Analysis)**.

3.2 Desarrollo

3.2.1 Requisitos Normativos Detallados del Sistema de Aire Acondicionado (CS-25)

La normativa **CS-25** [European Union Aviation Safety Agency, 2007] (*Certification Specifications 25*) es el código de aeronavegabilidad de la Agencia de Seguridad Aérea de la Unión Europea (EASA). Este marco regulatorio se aplica de forma mandatoria a aeronaves grandes, definidas como aquellas con un peso máximo certificado al despegue (**MTOW**) superior a los 5.700 kg (12.500 libras) y propulsadas por motores de turbina.

La aeronave objeto del presente proyecto, con un MTOW en su versión militar más simple de 23.200 kg y equipada con dos motores turbohélice, satisface ambos criterios, por lo que el cumplimiento de la **CS-25** es un requisito fundamental para el desarrollo del sistema de aire acondicionado propuesto.

La estructura de la **CS-25** se divide en dos volúmenes principales:

- **Libro 1: Especificaciones de Certificación.** Este primer volumen compila el cuerpo normativo esencial que rige tanto el diseño como la operación de las aeronaves.
- **Libro 2: Medios Aceptables de Cumplimiento (AMC).** El segundo volumen presenta los *Acceptable Means of Compliance* (AMC). Estos AMC no constituyen requisitos adicionales, sino que ofrecen directrices y metodologías validadas por la EASA para demostrar la conformidad.

Normativa general aplicable a equipos y sistemas embarcados (CS 25.1309)

A continuación, se presenta el texto de la normativa general aplicable a todo sistema o equipo embarcado.

> **CS 25.1309 EQUIPMENT, SYSTEMS AND INSTALLATIONS**
>
> (a) The aeroplane equipment and systems must be designed and installed so that: (1) Those required for type certification or by operating rules, or whose improper functioning would reduce safety, perform as intended under the aeroplane operating and environmental conditions. (2) Other equipment and systems are not a source of danger in themselves and do not adversely affect the proper functioning of those covered by sub-paragraph (a)(1) of this paragraph.
>
> (b) The aeroplane systems and associated components, considered separately and in relation to other systems, must be designed so that - (1) Any catastrophic failure condition (i) is extremely improbable; and (ii) does not result from a single failure; and (2) Any hazardous failure condition is extremely remote; and (3) Any major failure condition is remote.
>
> (c) Information concerning unsafe system operating conditions must be provided to the crew to enable them to take appropriate corrective action. A warning indication must be provided if immediate corrective action is required. Systems and controls, including indications and annunciations must be designed to minimise crew errors, which could create additional hazards.
>
> (d) Electrical wiring interconnection systems must be assessed in accordance with the requirements of CS 25.1709.

Se pueden destacar las condiciones básicas y genéricas que deben cumplir los equipos y sistemas en cuanto al diseño e instalación: deben funcionar y cumplir su cometido bajo las condiciones de operación y ambientales de la aeronave. Aquí es interesante exponer la importancia del sistema que en este proyecto se diseña, pues será la base de la operación de los demás equipos y sistemas embarcados. Si el sistema de aire acondicionado y presurización no cumple con su cometido de forma adecuada, no se tendrán las condiciones de operación necesarias para el resto de sistemas.

Por otro lado, se hace referencia a la probabilidad y los efectos de fallo de los sistemas. En este proyecto se va a abordar cierto estudio de fallos y errores puntual para plantear redundacias de acuerdo a la fiabilidad que exigen los requisitos.

Presurización y sistema neumático de baja presión (CS 25.1438)

Esta sección hace referencia concreta a los sistemas de presurización y sistemas que se sirven de aire a baja presión.

> **CS 25.1438 PRESSURISATION AND LOW PRESSURE PNEUMATIC SYSTEMS**
>
> Pneumatic systems (ducting and components) served by bleed air, such as engine bleed air, air conditioning, pressurisation, engine starting and hotair ice-protection systems, which are essential for the safe operation of the aeroplane or whose failure may adversely affect any essential or critical part of the aeroplane or the safety of the occupants, must be so designed and installed as to comply the CS 25.1309 In particular account must be taken of bursting or excessive leakage. (See AMC 25.1438 paragraph 1 for strength and AMC 25.1438 paragraph 2 for testing).

En este extracto se hace referencia a los componentes y conductos empleados en los sistemas que se abordan. Se tratan como sistemas esenciales para la operación segura de la aeronave y cuyo fallo puede ser crítico para los ocupantes. Particularmente, se llama al capítulo del AMC en relación a escapes y fugas.

AMC 25.1438 Pressurisation and Low Pressure Pneumatic Systems

1 Strength

1.1 Compliance with CS 25.1309(b) in relation to leakage in ducts and components will be achieved if it is shown that no hazardous effect will result from any single burst or excessive leakage.

1.2 Each element (ducting and components) of a system, the failure of which is likely to endanger the aeroplane or its occupants, should satisfy the most critical conditions of Table 1.

[Nota: La Tabla 1 se describe a través de las siguientes variables]

- P1 = the most critical value of pressure encountered during normal functioning.
- T1 = the combination of internal and external temperatures which can be encountered in association with pressure P1.
- P2 = the most critical value of pressure corresponding to a probability of occurrence 'reasonably probable'.
- T2 = the combination of internal and external temperatures which can be encountered in association with pressure P2.

1.3 After being subjected to the conditions given in column 1 of Table 1, and on normal operating conditions being restored, the element should operate normally and there should be no detrimental permanent distortion.

1.4 The element should be capable of withstanding the conditions given in column 2 of Table 1 without bursting or excessive leakage.

2 Tests

2.1 Static tests. Each element examined under 1.2 should be static-tested to show that it can withstand the most severe conditions derived from consideration of the temperatures and pressures given in the Table.

2.2 Endurance tests. When failures can result in hazardous conditions, elements and/or sub-systems should be fatigue-tested under representative operating conditions that simulate complete flights to establish their lives.

Normativa aplicable a la ventilación de cabina (CS 25.831)

Esta normativa gobierna el suministro de aire que se debe aportar en cabina.

CS 25.831 Ventilation

(a) Each passenger and crew compartment must be ventilated and each crew compartment must have enough fresh air (but not less than 0.28 m^3/min. (10 cubic ft per minute) per crewmember) to enable crewmembers to perform their duties without undue discomfort or fatigue.

(b) Crew and passenger compartment air must be free from harmful or hazardous concentrations of gases or vapours. In meeting this requirement, the following apply: (1) Carbon monoxide concentrations in excess of one part in 20 000 parts of air are considered hazardous. (2) Carbon dioxide concentration during flight must be shown not to exceed 0.5 % by volume (sea level equivalent) in compartments normally occupied by passengers or crewmembers.

(c) There must be provisions made to ensure that the conditions prescribed in sub-paragraph (b) of this paragraph are met after reasonably probable failures or malfunctioning of the ventilating, heating, pressurisation or other systems and equipment.

(d) If accumulation of hazardous quantities of smoke in the cockpit area is reasonably probable, smoke evacuation must be readily accomplished.

> (e) [...] means must be provided to enable the occupants of the [...] flight-crew compartment [...] to control the temperature and quantity of ventilating air supplied to their compartment or area independently of the temperature and quantity of air supplied to other compartments and areas.

Se determina el mínimo caudal de aire fresco ($0.28\ m^3/min$ por tripulante), se limita la concentración de monóxido de carbono (< 1 parte en 20,000) y dióxido de carbono (< 0.5 %), y se exige la capacidad de controlar la temperatura y ventilación de forma independiente para la tripulación.

Normativa aplicable a la presurización de cabina (CS 25.841 y CS 25.843)

> **CS 25.841 Pressurised cabins**
>
> (a) Pressurised cabins and compartments to be occupied must be equipped to provide a cabin pressure altitude of not more than 2438 m (8000 ft) at the maximum operating altitude of the aeroplane under normal operating conditions. If certification for operation over 7620 m (25 000 ft) is requested, the aeroplane must be able to maintain a cabin pressure altitude of not more than 4572 m (15 000 ft) in the event of any reasonably probable failure or malfunction in the pressurisation system.
>
> (b) Pressurised cabins must have at least the following valves, controls, and indicators for controlling cabin pressure: (1) Two pressure relief values... (2) Two reverse pressure differential relief valves... (3) A means by which the pressure differential can be rapidly equalised. (4) An automatic or manual regulator... (5) Instruments at the pilot or flight engineer station... (6) Warning indication at the pilot or flight engineer station...

Las cabinas deben mantener una altitud de presión máxima de 8000 ft en operación normal, y de 15000 ft en caso de fallo probable si se opera por encima de 25000 ft. Se lista el equipamiento mínimo de control (válvulas, reguladores, indicadores y alarmas).

> **CS 25.843 Tests for pressurised cabins**
>
> (a) Strength test. The complete pressurized cabin, including doors, windows, and valves, must be tested as a pressure vessel for the pressure differential specified in CS 25.365 (d).
>
> (b) Functional tests. The following functional tests must be performed: (1) Tests of the functioning and capacity of the positive and negative pressure differential valves... (2) Tests of the pressurisation system to show proper functioning under each possible condition... (3) Flight tests, to show the performance of the pressure supply, pressure and flow regulators... (4) Tests of each door and emergency exit, to show that they operate properly after being subjected to the flight tests...

Se definen las pruebas requeridas: una de resistencia estructural (Strength test) y varias pruebas funcionales para validar el comportamiento del sistema, sus componentes y accesos en todas las condiciones de vuelo.

Normativa aplicable a concentración de ozono en cabina (CS 25.832)

> **CS 25.832 Cabin ozone concentration**
>
> (a) The aeroplane cabin ozone concentration during flight must be shown not to exceed – (1) 0.25 parts per million by volume, sea level equivalent, at any time above flight level 320; and (2) 0.1 parts per million by volume, sea level equivalent, time-weighted average during any 3-hour interval above flight level 270.
>
> (c) Compliance with this paragraph must be shown by analysis or tests...

Se establecen límites estrictos para la concentración de ozono (O_3) en cabina para garantizar la seguridad y habitabilidad.

Normativa aplicable a la temperatura

Temperatura en cabina (Requisitos de Diseño Térmico)

Se definen los requisitos para los casos más desfavorables de calor y frío.

> **Thermal design requirements**
>
> - **Worst cooling case (hottest case):** Cooling on ground at a hot and humid place, aircraft full, doors closed, APU power. Refrigeration must be able to cool from 47°C to 21°C in <30 min.
> - **Worst heating case (coldest case):** Heating on ground at a cold and humid place, aircraft empty, doors closed, APU power. Heating must be able to heat from -40°C to 24°C in <30 min.

Estos requisitos operacionales de enfriamiento y calentamiento condicionarán y dimensionarán el diseño del sistema.

Temperatura atmosférica (Pruebas de enfriamiento)

Se definen las condiciones atmosféricas para las pruebas de certificación.

> **CS 25.1043 Cooling tests**
>
> (b) Maximum ambient atmospheric temperature. A maximum ambient atmospheric temperature corresponding to sea level conditions of at least 37.8°C (100°F) must be established. The assumed temperature lapse rate is 6.6°C per thousand meter (3.6°F per thousand feet) of altitude above sea level until a temperature of -56.5°C (–69.7°F) is reached...
>
> **CS 25.1045 Cooling test procedures**
>
> (a) Compliance with CS 25.1041 must be shown for the take-off, climb, en-route, and landing stages of flight that correspond to the applicable performance requirements. The cooling tests must be conducted with the aeroplane in the configuration, and operating under the conditions, that are critical relative to cooling during each stage of flight.

Toma de Aire (CS 25.1091)

Requisitos relevantes para el sistema de sangrado que alimenta el ACS.

> **CS 25.1091 Air intake**
>
> (a) The air intake system for each engine must supply – (1) The air required by that engine under each operating condition for which certification is requested...
>
> (d) [...] The aeroplane must be designed to prevent water or slush on the runway, taxiway, or other airport operating surfaces from being directed into the engine air intake ducts in hazardous quantities...
>
> (e) If the engine air intake system contains parts or components that could be damaged by foreign objects entering the air intake, it must be shown by tests or, if appropriate, by analysis that the air intake system design can withstand the foreign object ingestion test conditions...

3.3 Uso de IA para procesar los Requisitos e introducirlos en Capella

Una vez localizados los requisitos normativos, se empleó un flujo de trabajo automatizado para su gestión. Primero, mediante un script de MATLAB (ver Apéndice A), y asistido por herramientas de IA generativa para la estructuración inicial, los requisitos fueron importados y organizados jerárquicamente en la **Requirement Toolbox** (**Figura 3.1**). Esta herramienta, compatible con estándares industriales como **IBM DOORS**, permitió generar un fichero de intercambio estándar **ReqIF**.

Posteriormente, desde **Capella**, se importó dicho fichero utilizando el *add-on* **Requirements Viewpoint**, vinculando así de forma trazable la normativa directamente a las distintas capas de la arquitectura del modelo (**Figuras 3.2**, 3.3 y 3.4). Este procedimiento no solo agiliza el proceso, sino que garantiza una *fuente única y consistente* para los requisitos del sistema.

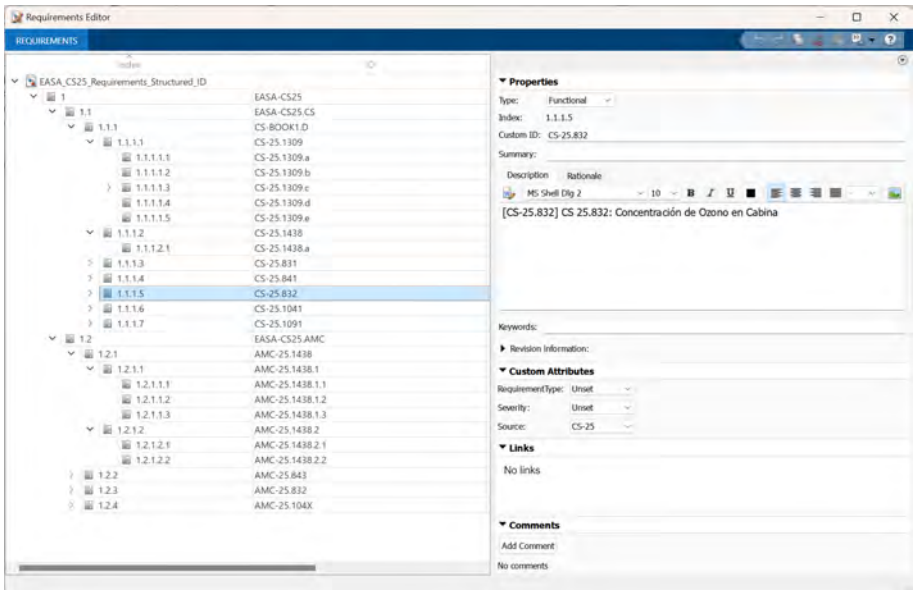

Figura 3.1 Ejemplo de estructura de requisitos en MATLAB Requirement Toolbox.

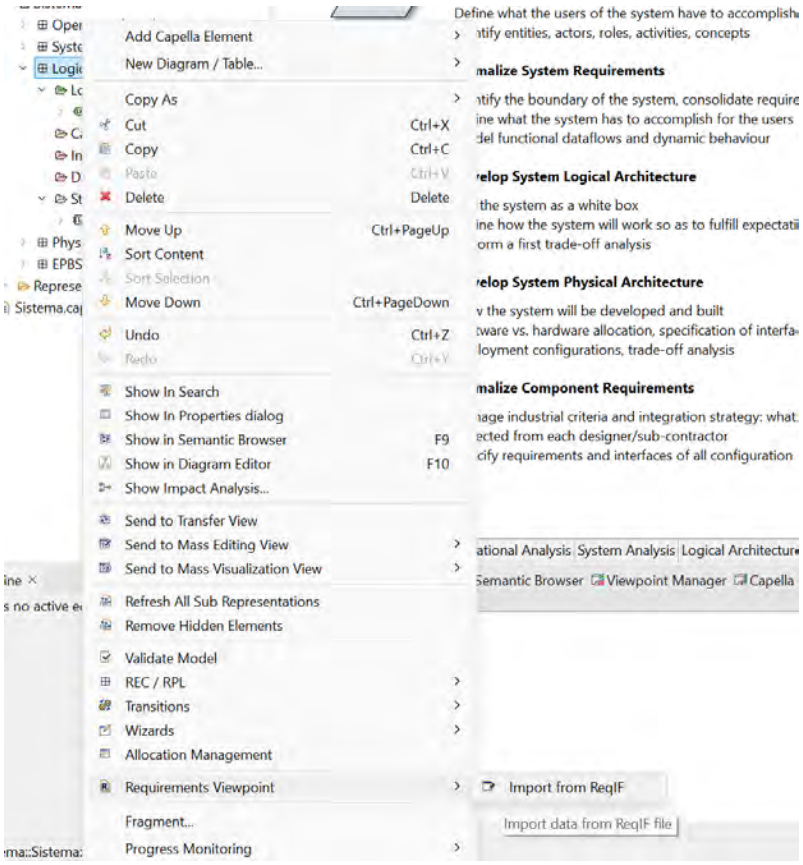

Figura 3.2 Selección de la capa para añadir requisitos en Capella.

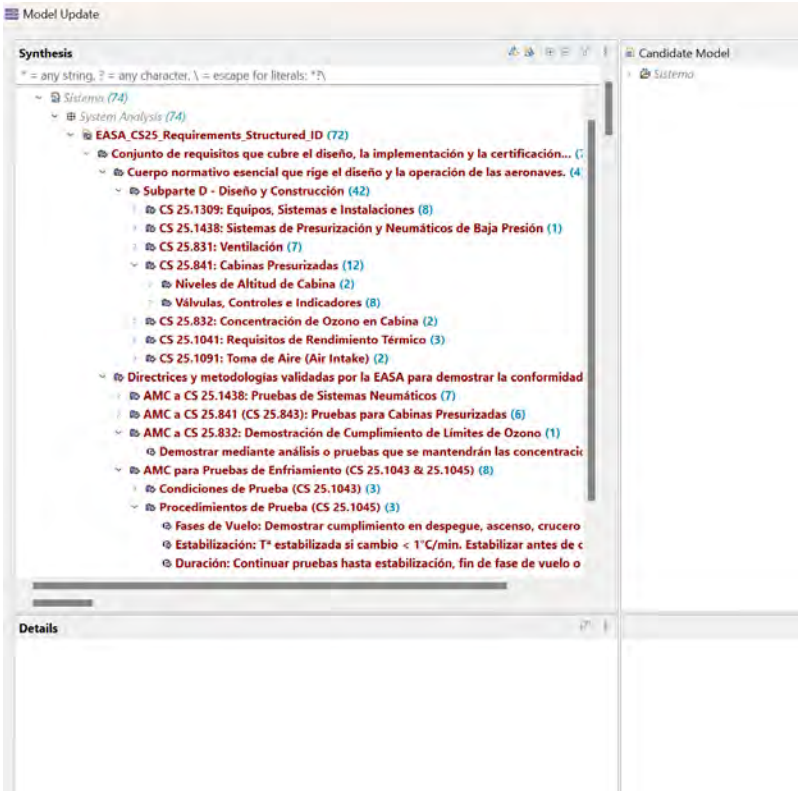

Figura 3.3 Estructura jerárquica de requisitos de aeronavegabilidad (EASA CS-25).

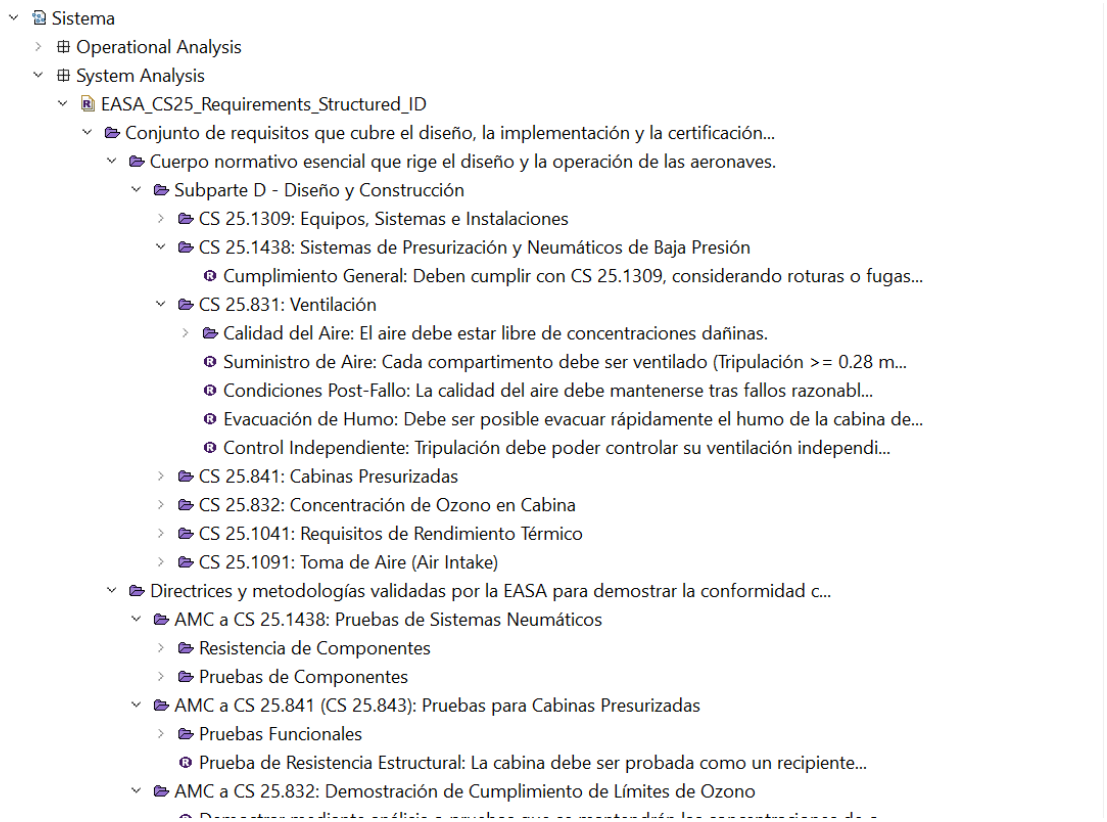

Figura 3.4 Ejemplo de descomposición de requisitos para sistemas de cabina dentro de Capella.

3.4 Fundamentos Técnicos para el desarrollo de un Sistema de Presurización y Aire Acondicionado, ATA 21.

3.4.1 El Sistema de presurización y Aire Acondicionado de Cabina, ATA 21

Introducción y Principios Fundamentales

El Sistema de presurización y aire acondicionado, es uno de los sistemas más críticos para la operación de aeronaves comerciales. Su misión es mantener un entorno habitable en el interior del fuselaje a altitudes de vuelo donde la atmósfera exterior es letal por su baja presión, baja temperatura y alta concentración de ozono.

Las tres funciones primordiales del ECS son:

- **Presurización:** Mantener una presión en cabina equivalente a una altitud fisiológicamente segura para prevenir la hipoxia.
- **Control de Temperatura:** Regular la temperatura de la cabina para el confort de los ocupantes y la correcta operación de la aviónica, gestionando fuentes de calor extremas.
- **Ventilación:** Asegurar la renovación y pureza del aire, controlando los niveles de ozono y otros contaminantes, y eliminando olores.

Para cumplir estos objetivos, el ECS se fundamenta en los principios de la termodinámica y la mecánica de fluidos, utilizando como fuente de energía principal el aire sangrado de los motores.

Fuente de Potencia Neumática y de Aire de Sangrado: El Sistema de Aire de Sangrado

La energía necesaria para el ECS se obtiene neumáticamente del sistema de aire de sangrado (**Bleed Air System**). Este sistema extrae una fracción del aire que fluye a través de los compresores de los motores principales antes de que dicho aire llegue a la cámara de combustión.

Puntos de Extracción: El aire se extrae de puertos situados en las etapas de presión intermedia (**IP** - Intermediate Pressure) o alta (**HP** - High Pressure) del compresor del motor. Un sistema de control selecciona automáticamente el puerto HP solo cuando la potencia del motor es baja y la presión en el puerto IP es insuficiente. En tierra, el aire es suministrado por la Unidad de Potencia Auxiliar (**APU**) o directamente de una fuente externa de aire acondicionado. En la **Figura 3.5** se representa el proceso descrito para dos motores.

Propiedades del Aire de Sangrado: Al ser extraído tras varias etapas de compresión, este aire se encuentra en condiciones extremas: su temperatura puede superar los 200°C y su presión es del orden de 40 psi.

Regulación y Acondicionamiento: Antes de ser utilizado, el aire de sangrado debe ser regulado.

- Un Controlador de Monitorización de Sangrado (**Bleed Monitoring Computer - BMC**) supervisa el sistema.
- Una Válvula Reguladora y de Corte de Presión (**PRSOV**) modula el flujo para estabilizar la presión y temperatura de suministro.
- Un pre-enfriador (**pre-cooler**), que es un intercambiador de calor aire-aire, utiliza aire frío del fan del motor (**Fan Air**) para reducir la temperatura del aire de sangrado y proteger los conductos y componentes aguas abajo.
- Un convertidor catalítico de ozono descompone el O_3 en O_2 para garantizar que el aire sea respirable según la normativa.

Este aire regulado no solo alimenta los PACKs de aire acondicionado, sino que también es distribuido para funciones como el antihielo de alas (**WAI**), la inertización de tanques de combustible (**OBIGGS**) y el arranque de motores.

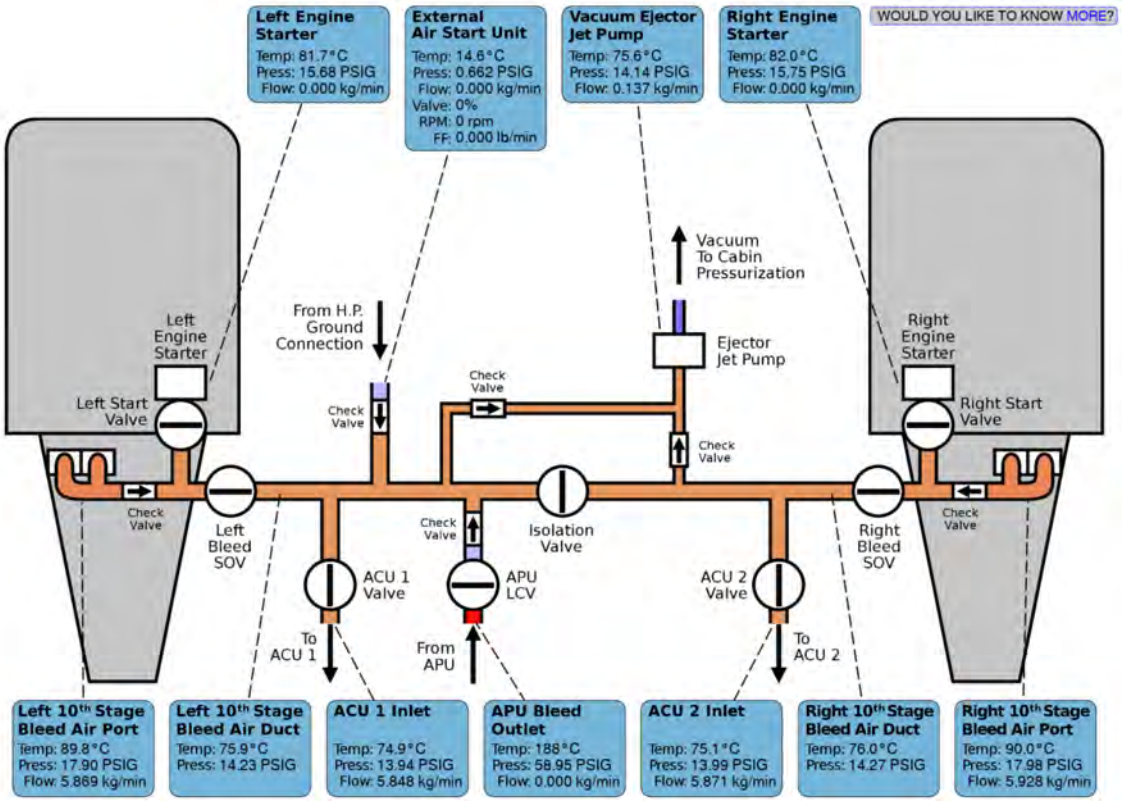

Figura 3.5 Diagrama del sistema neumático de una aeronave.

El Ciclo de Refrigeración: Análisis del PACK de Aire Acondicionado (ACP)

El núcleo del ECS es el **PACK de Aire Acondicionado (Air Conditioning PACK - ACP)**. Las aeronaves comerciales están equipadas con dos unidades redundantes. Cada PACK es una máquina termodinámica que ejecuta un ciclo Brayton inverso para producir aire a temperaturas extremadamente bajas a partir del aire de sangrado caliente. El disipador de calor para este ciclo es el aire de impacto (**Ram Air**), que es aire exterior capturado por unas tomas dinámicas. El proceso termodinámico dentro del PACK se desarrolla en las siguientes etapas:

1. **Primer Intercambiador de Calor (PHE):** El aire de sangrado caliente cede una parte inicial de su energía térmica al flujo de Ram Air.
2. **Compresor de la ACM:** El aire, ya preenfriado, entra en el compresor de la **Máquina de Ciclo de Aire (ACM)**. Su presión y temperatura se incrementan notablemente. Este paso es termodinámicamente crucial: al aumentar la temperatura del fluido de trabajo, se maximiza el gradiente de temperatura (ΔT) con el refrigerante (Ram Air) en la siguiente etapa, lo que aumenta drásticamente la eficiencia de la transferencia de calor según la ley de enfriamiento de Newton.
3. **Segundo Intercambiador de Calor (SHE):** El aire, ahora a su máxima temperatura, pasa por el intercambiador principal, donde se produce la mayor parte de la disipación de calor hacia el flujo de Ram Air.
4. **Turbina de la ACM:** El aire, a alta presión pero con una temperatura ya considerablemente reducida, se expande a través de la turbina de la ACM. Esta expansión cuasi-adiabática provoca una drástica caída de la temperatura (hasta valores bajo cero) al convertir la entalpía del aire en trabajo mecánico. Este trabajo se transmite a través de un eje para accionar el compresor de la etapa 2, haciendo el ciclo autosuficiente y energéticamente eficiente.

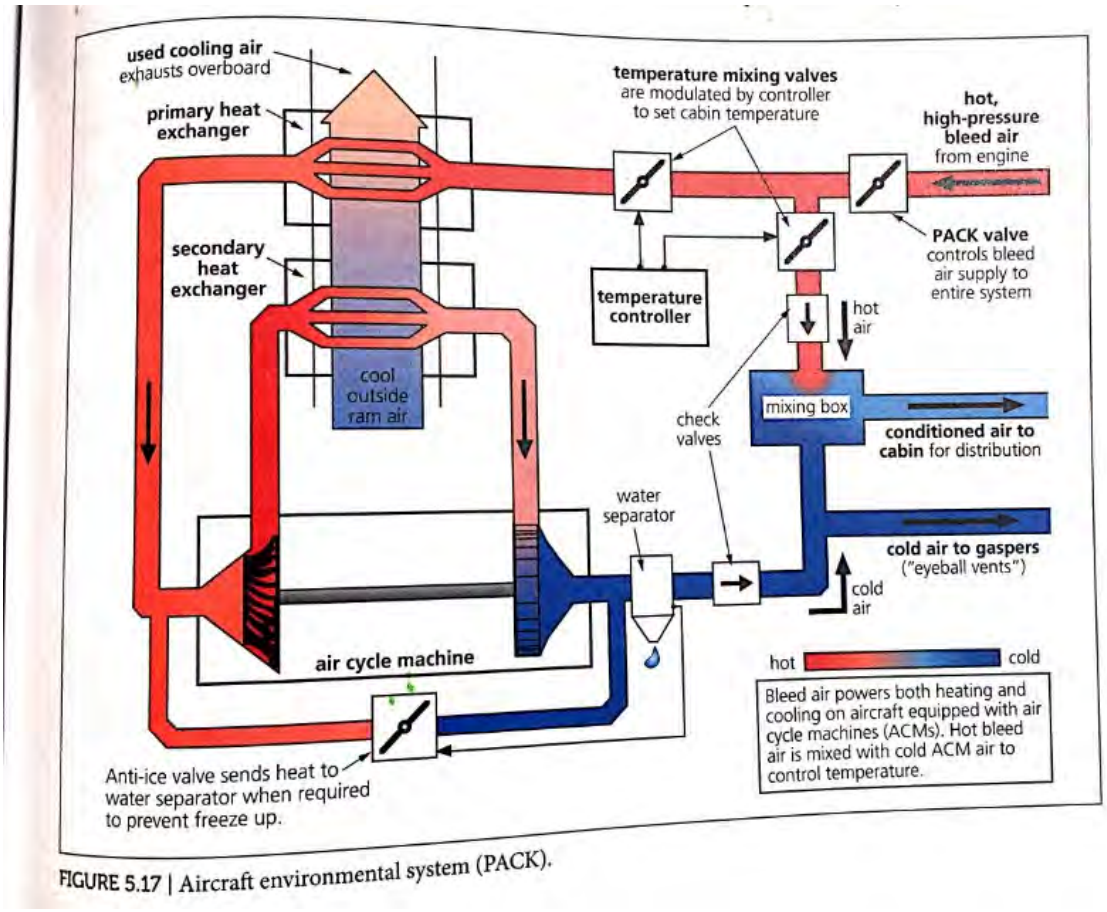

Figura 3.6 Esquema detallado de un PACK del Sistema de Control Ambiental (ECS). Fuente: *Aircraft Systems for Pilots*.

5. **Separador de Agua:** El aire a la salida de la turbina se encuentra por debajo de su punto de rocío, provocando la condensación y congelación de la humedad. Un separador de agua de tipo ciclónico extrae estas partículas de agua líquida/hielo para evitar la introducción de niebla en la cabina.

Para mejor entendimiento mirar la **Figura 3.6** que representa un esquema detallado del ECS.

Control de Temperatura y Distribución en Cabina

El aire que emerge del ACP es demasiado frío para ser suministrado directamente.

Control de Temperatura del PACK: Una **Válvula de Control de Temperatura (TCV)** toma una pequeña cantidad de aire de sangrado caliente y la "bypasea" alrededor del PACK, mezclándola con el aire gélido a la salida de la turbina. La modulación de esta válvula permite un control preciso de la temperatura de salida del PACK. Se puede vislumbrar su posición en el sistema en la **Figura 3.7**.

Unidad de Mezcla y Recirculación: El aire acondicionado de los PACKs se dirige a una cámara de mezcla (**mixing chamber**). Aquí, se combina con el aire de recirculación de la cabina. Aproximadamente el 50 % del flujo que se distribuye es aire de la propia cabina que ha sido aspirado y filtrado a través de filtros **HEPA (High-Efficiency Particulate Air)**, que eliminan partículas, bacterias y virus con una altísima eficiencia. Esta recirculación reduce la carga sobre los motores, ahorrando combustible.

Control de Temperatura Zonal (Trim Air System): Para satisfacer las diferentes necesidades térmicas de la aeronave (cockpit vs. cabina de pasajeros), el sistema de distribución final permite

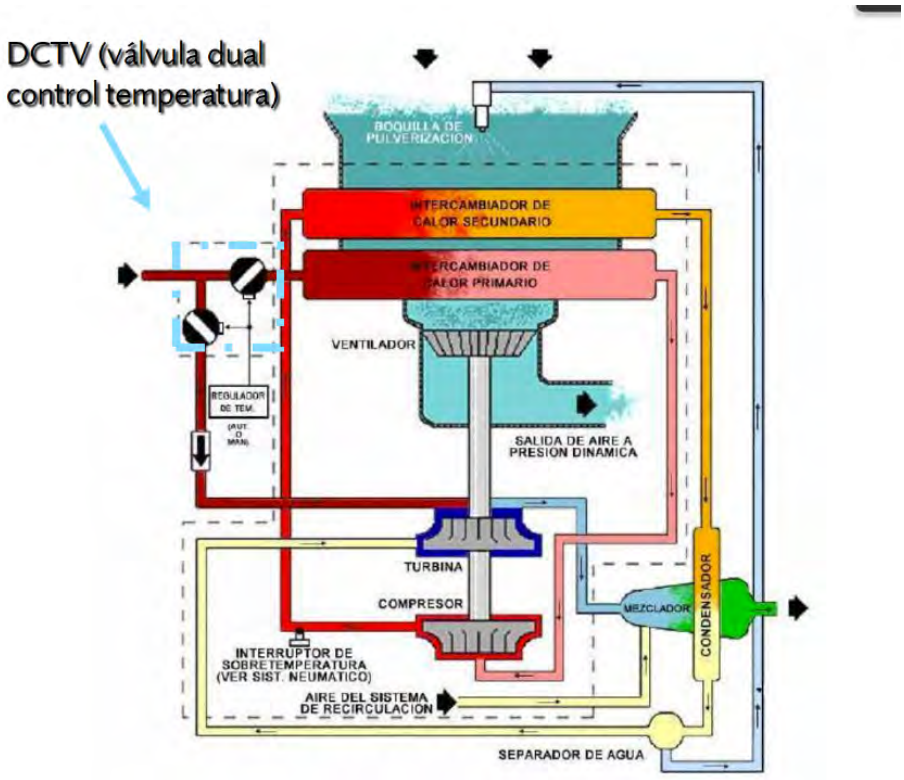

Figura 3.7 Diagrama de una Máquina de Ciclo de Aire (ACM) y sus componentes principales dentro de un PACK de aire acondicionado. Muestra la disposición de las Valvulas DCTV.

un ajuste zonal. Se inyectan pequeñas cantidades de aire caliente de sangrado (**Trim Air**) en los conductos de cada zona a través de unas **válvulas de ajuste (Hot Air Trim Valves)**, modificando localmente la temperatura final.

Sistema Doble de Presurización Automática

El flujo de entrada de los PACKs debe ser relativamente constante.

Controladores de Presión (CPCs): El sistema es gestionado por dos **Controladores de Presión de Cabina (CPCs)** redundantes. Estos reciben información del sistema de gestión de vuelo (**FMS**), como la altitud de crucero y la altitud del aeropuerto de destino, y datos de los sensores de presión. Con esta información, calculan un perfil de presurización para todo el vuelo.

Válvula de Salida (Outflow Valve): El CPC actúa sobre una o varias **válvulas de salida (outflow valves)**. Estas grandes válvulas, normalmente localizadas en la parte trasera inferior del fuselaje, modulan su apertura para regular la cantidad de aire que se expulsa de la cabina, manteniendo así la presión interna en el valor dictado por el controlador.

Protecciones de Seguridad: El fuselaje está protegido por válvulas de alivio mecánicas que actúan independientemente del sistema de control para prevenir una condición de sobrepresión o de presión negativa diferencial que pudiera comprometer la integridad estructural de la aeronave.

Interfaz de Operación y Supervisión

Dentro de lo que se conoce como **Sistema de instrumentación (ATA 31)**. La tripulación de vuelo interactúa con el ECS a través del panel superior (**Overhead Panel**), donde pueden seleccionar los modos de operación (normalmente AUTO) y las temperaturas deseadas para cada zona. La supervisión del sistema se realiza a través de las pantallas **ECAM (Electronic Centralized Aircraft Monitor)** o **EICAS (Engine Indicating and Crew Alerting System)**. Estas presentan páginas sinópticas que muestran en tiempo real el estado de las válvulas, las temperaturas y presiones en

puntos clave del sistema, y la información de presurización (**Figura 3.8**, **Figura 3.9** y **Figura 3.10**). En caso de una anomalía, el sistema genera alertas y advertencias, guiando a la tripulación en la resolución del problema.

Figura 3.8 Pantalla del sistema de presurización de cabina (CAB PRESS) en ECAM.

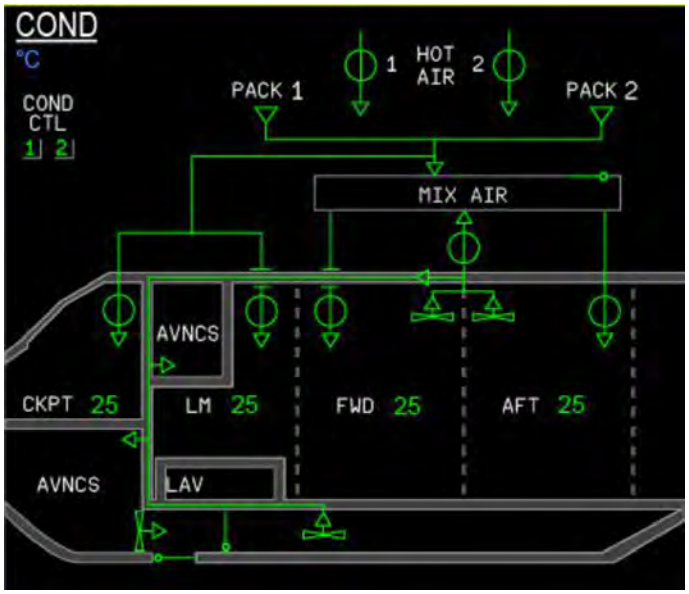

Figura 3.9 Diagrama del sistema de aire acondicionado (ECS) en ECAM.

Una vez definido de manera general el sistema objeto de estudio, así como aquellos con los que interactúa más cercanamente para proveer de sus capacidades, se tendrá que exponer cómo es el proceso que implica el diseño de este subsistema de un avión. Lejos de ser algo dado, responde a un desarrollo mediante hipótesis iniciales, interacción y colaboración entre departamentos, refinamiento de los métodos utilizados, iteración y finalmente convergencia de los resultados. Cabe señalar que en nuestro caso se trabajará con una arquitectura ya más o menos prefigurada, cosa que no ocurriría si se tratase de diseñar un nuevo sistema de cero.

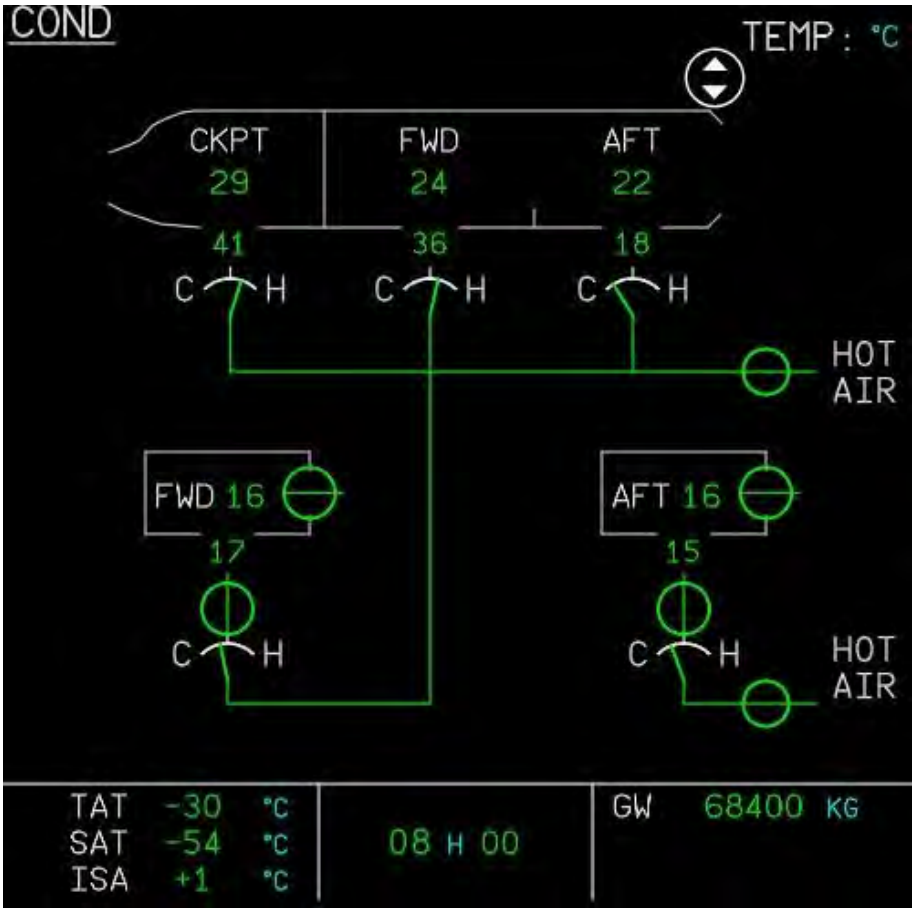

Figura 3.10 Distribución de temperaturas por zona en la cabina de una aeronave en ECAM.

3.5 Aplicación de la Metodología ARCADIA

3.5.1 Análisis Operacional

Siguiendo la metodología ARCADIA, el Análisis Operacional es el primer paso y se enfoca en comprender el contexto en el que el sistema operará, identificando las necesidades de los usuarios y otros sistemas externos. Se trata al sistema ACPC (Air Conditioning and Pressurization Control) como una "caja negra" y se fija en lo que se espera de él desde una perspectiva externa.

Identificación de Actores Operacionales (Stakeholders)

Los actores operacionales son las entidades (personas, organizaciones u otros sistemas) que interactúan con el sistema ACPC o tienen un interés en su funcionamiento[1]. Para nuestro sistema, se identifican los siguientes actores principales:

Tripulación de Vuelo (Pilotos):

- Necesitan controlar y monitorizar el sistema.
- Necesitan que el sistema mantenga la cabina presurizada adecuadamente según la altitud de vuelo.

Pasajeros/Personal Transportado:

[1] Las Autoridades de Certificación Aeronáutica (ej. EASA, FAA) y el Fabricante de la Aeronave (Airbus Defence and Space), aunque son stakeholders cruciales, no se incluirán en el análisis detallado posterior para acotar el alcance del trabajo.

- Necesitan un ambiente confortable en términos de temperatura, calidad del aire (ventilación, ausencia de olores) y presión.
- Su bienestar y capacidad para operar (en caso de personal militar o misiones especiales) dependen de un ambiente adecuado.

Tripulación de Cabina/Auxiliares de Vuelo (si aplica según configuración):

- Al igual que los pasajeros, necesitan un ambiente confortable.
- Pueden tener algún nivel de control o solicitud sobre la temperatura de la zona de pasajeros.

Personal de Mantenimiento en Tierra:

- Necesitan acceder al sistema para inspecciones, mantenimiento preventivo y correctivo.
- Necesitan herramientas de diagnóstico y la capacidad de probar el sistema en tierra.
- Necesitan que el sistema facilite la identificación y solución de fallos.

Necesidades Operacionales de los Actores

A partir de los actores identificados, se deriban sus necesidades fundamentales con respecto al sistema ACPC. **Necesidades de Confort y Ambientales (Comunes a Tripulación y Pasajeros)**

N1: Necesidad de una presión de cabina segura y confortable en todas las fases del vuelo.

N2: Necesidad de que la temperatura de la cabina se mantenga dentro de un rango confortable.

N3: Necesidad de una ventilación adecuada y renovación del aire.

Necesidades de Control y Supervisión (Específicas de la Tripulación de Vuelo)

N4: Necesidad de ser informado sobre el estado y posibles fallos del sistema ACPC.

N5: Necesidad de actuar sobre los controles del sistema.

Necesidades de Mantenimiento (Específicas del Personal de Tierra)[2]

N6: Necesidad de diagnosticar fallos del sistema de forma eficiente.

N7: Necesidad de realizar pruebas funcionales del sistema en tierra.

N8: Necesidad de acceder físicamente a los componentes para su reparación o reemplazo.

Objetivos y Capacidades Operacionales (Operational Capabilities)

Las capacidades operacionales describen lo que el sistema ACPC debe ser capaz de hacer para satisfacer las necesidades de los actores. Estas capacidades operacionales y sus relaciones con los actores vienen dadas en la siguiente Figura:

[2] Más adelante se verá que se ha decidido no continuar con el desarrollo de estas necesidades relacionadas con el mantenimiento por limitaciones de tiempo para el desarrollo de este proyecto y por carecer de interés suficiente.

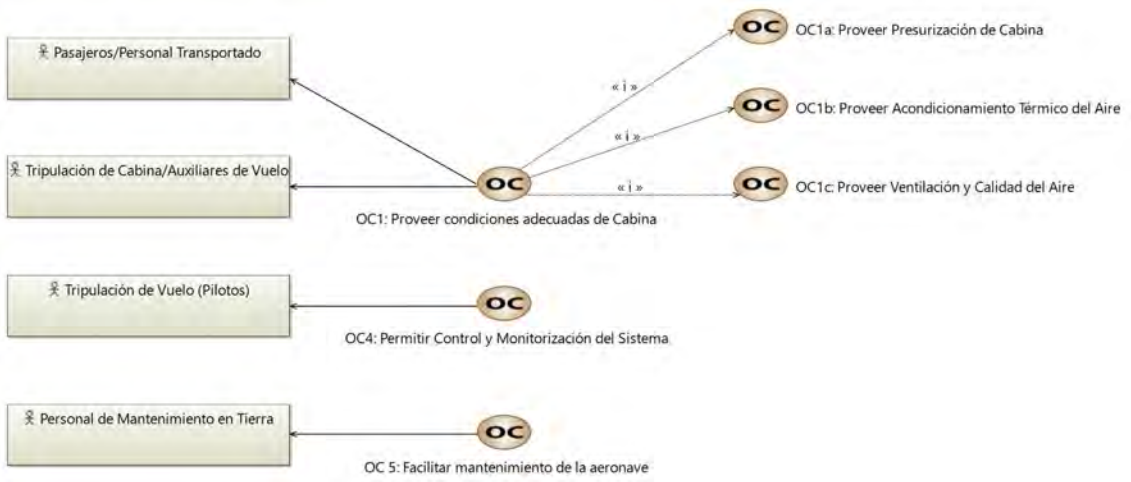

Figura 3.11 Esquema OCB.

OC1a: Proveer Presurización de Cabina *Responde a: **N1***. Mantener la altitud de cabina a un nivel seguro y confortable (típicamente no excediendo el equivalente a 8000 pies) durante el ascenso, crucero y descenso, y controlar la tasa de cambio de presión. Incluye la capacidad de presurización diferencial positiva y negativa controlada.

OC1b: Proveer Acondicionamiento Térmico del Aire *Responde a: **N2***. Calentar o enfriar el aire suministrado a la cabina y al cockpit para mantener una temperatura seleccionada y confortable, adaptándose a las condiciones externas e internas.

OC1c: Proveer Ventilación y Calidad del Aire *Responde a: **N3***. Asegurar una tasa de renovación de aire adecuada, filtrar el aire de partículas y olores, y controlar la humedad. Prevenir la entrada de contaminantes.

OC4: Permitir Control y Monitorización del Sistema[3] *Responde a: **N4, N5***. Proporcionar a la tripulación de vuelo interfaces para seleccionar modos de operación, ajustar parámetros y visualizar el estado del sistema, incluyendo alertas y advertencias.

OC5: Facilitar Mantenimiento del Sistema *Responde a: **N6, N7, N8***. Incorporar capacidades de auto-diagnóstico, permitir pruebas en tierra, y diseñar componentes para un acceso y reemplazo razonables.

Actividades Operacionales y Escenarios

Las actividades operacionales describen cómo los actores utilizan las capacidades del sistema en diferentes contextos.

Escenario: Vuelo Rutinario

Actor: Tripulación de Vuelo

- Monitorizar la presurización, temperatura y ventilación durante el ascenso, crucero y descenso (Usa **OC4**).
- Ajustar la temperatura, presurización y ventilación si es solicitado o si las condiciones cambian (Usa **OC1b, OC4**).

Actor: Pasajeros

- Experimentar un ambiente presurizado y con temperatura controlada (Se beneficia de **OC1a, OC1b, OC1c**).

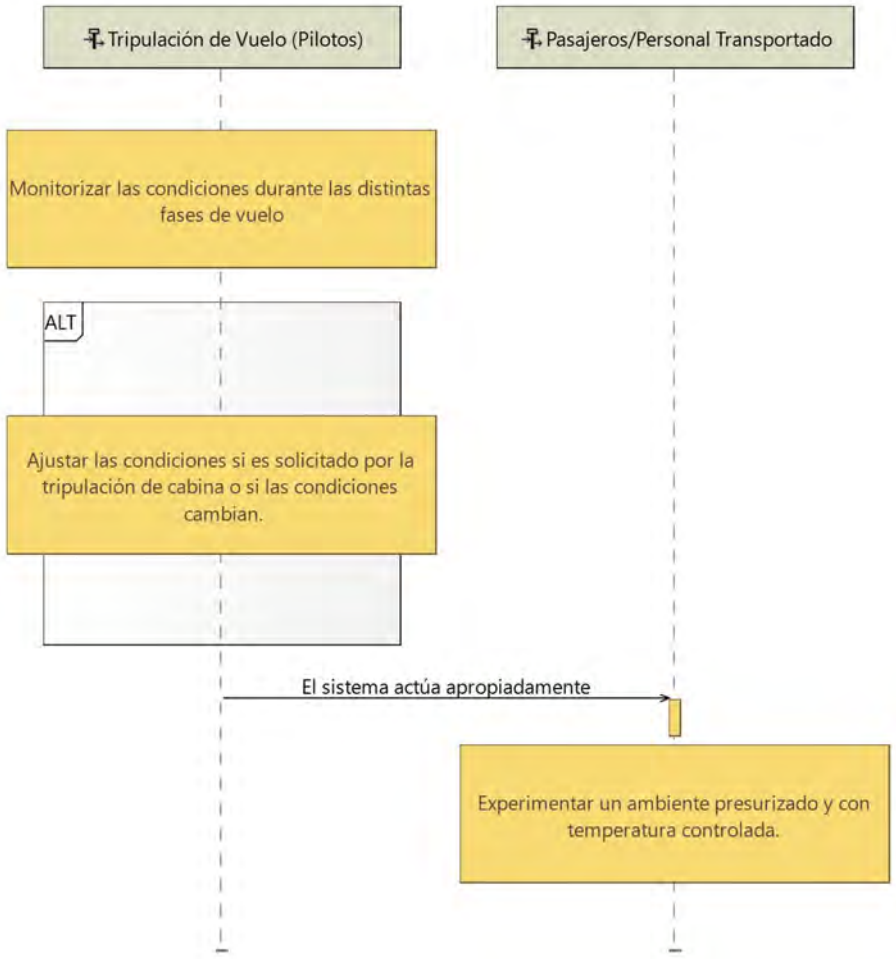

Figura 3.12 Esquema ES: Vuelo rutinario.

Escenario: Avión en Tierra

Actor: Tripulación de Vuelo

- Configurar el sistema ACPC (seleccionar modo AUTO, verificar presurización programada) (Usa **OC4**).
- Monitorizar la presurización, temperatura y ventilación en tierra (Usa **OC4**).
- Ajustar la temperatura, presurización y ventilación si es necesario (Usa **OC1b, OC4**).

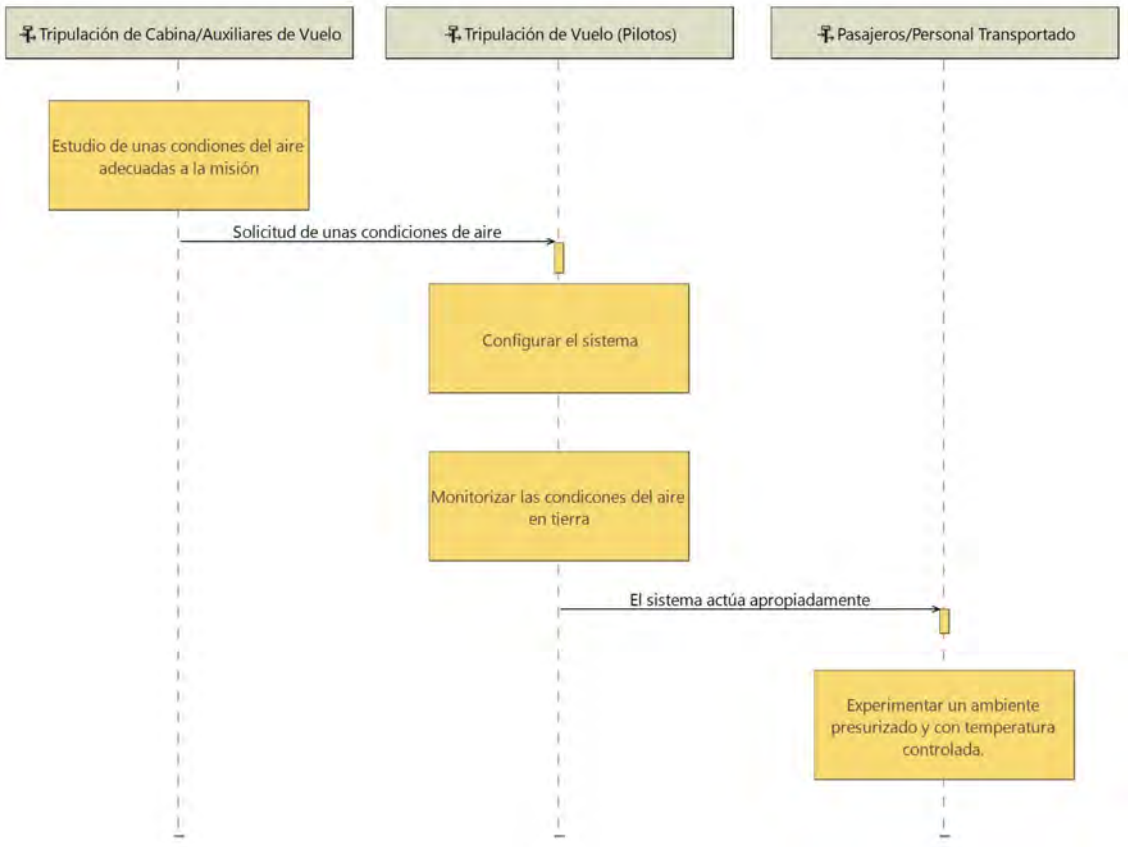

Figura 3.13 Esquema ES: Avión en Tierra.

Escenario: Mantenimiento Programado en Tierra

Actor: Personal de Mantenimiento

- Realizar pruebas funcionales del sistema ACPC usando el panel de mantenimiento (Usa **OC5**).
- Inspeccionar y reemplazar filtros de aire (Usa **OC5**).
- Verificar el estado de las válvulas y sensores (Usa **OC5**).

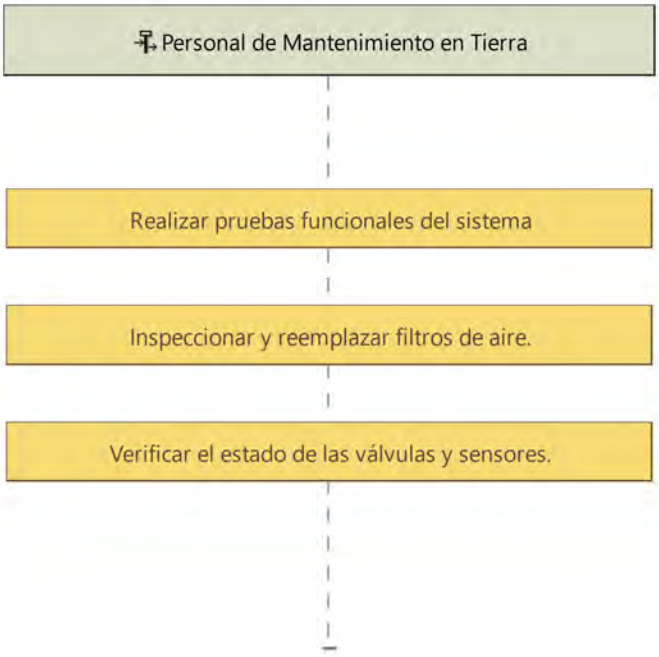

Figura 3.14 Esquema ES: Mantenimiento del sistema.

Consideraciones sobre el Alcance y la Modelización

El diagrama de arquitectura operacional [OAB] (**Figura 3.15**) se genera a partir de este análisis. Todas las interacciones y actividades previamente definidas se trasladan al modelo, asegurando que todos los diagramas (e.g., [OAB], [OES]) queden interconectados. Esta trazabilidad es una ventaja clave, ya que una modificación en una entidad (como el nombre de una actividad) se propaga automáticamente a través de todos los diagramas relevantes.

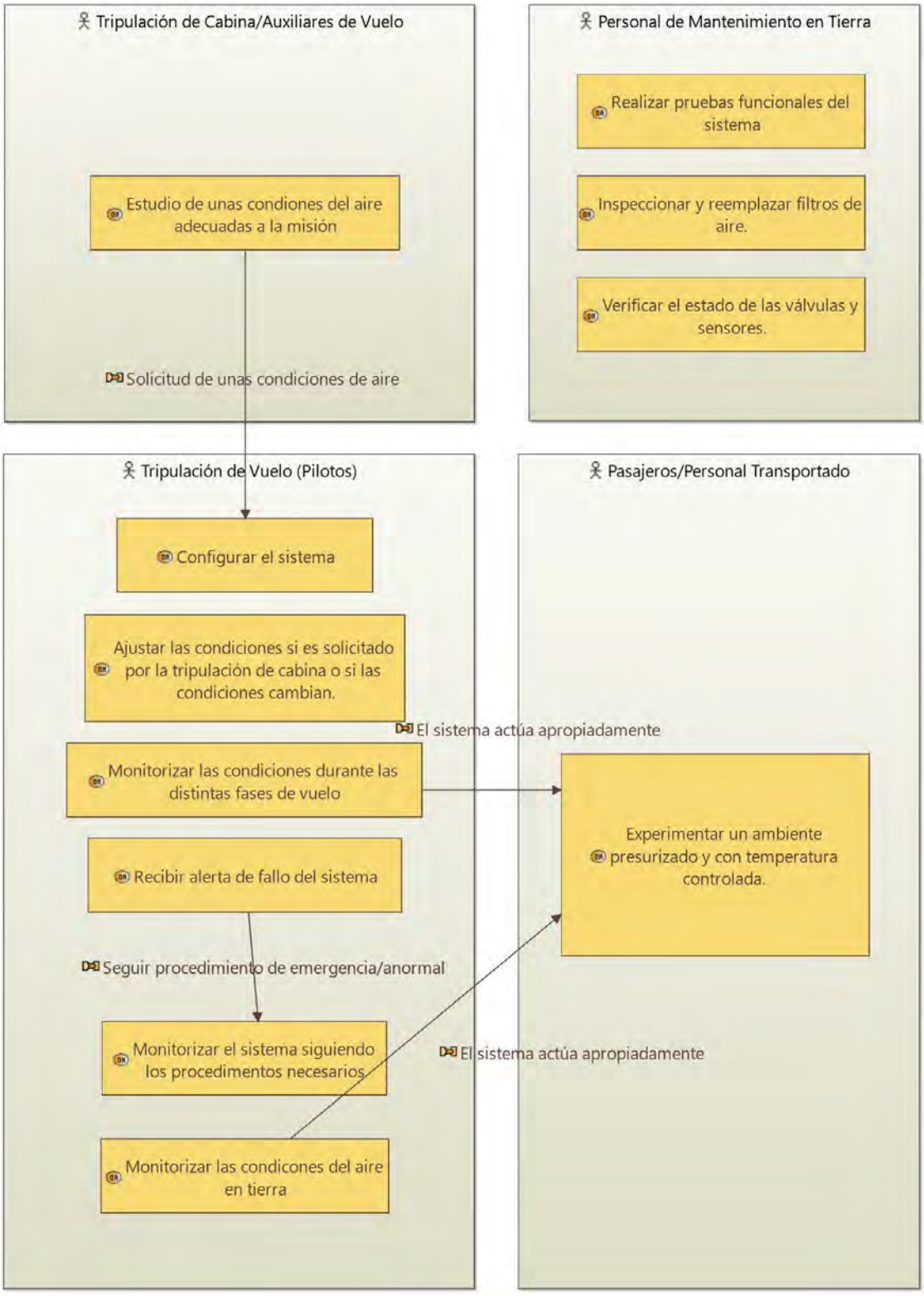

Figura 3.15 Esquema OAB.

Para mejorar la legibilidad del diagrama, se ha optado por acortar las etiquetas de las actividades. Por ejemplo, las referencias a "temperatura, presión y ventilación" se han consolidado bajo el término "condiciones". Asimismo, algunas actividades conceptuales se han modelado más adecuadamente

como interacciones entre actividades.

Se podría haber realizado un análisis mucho más extenso de las interacciones de los actores con el sistema. Sin embargo, un mayor nivel de detalle en esta fase inicial implicaría una complejidad excesiva para las fases posteriores, excediendo el alcance de este Trabajo de Fin de Grado (TFG). Por consiguiente, el análisis no se detendrá en escenarios de fallo complejos (como fallo de motor o de un PACK) ni en todas las actuaciones detalladas de los pilotos en tierra, ya que requerirían un tiempo de desarrollo desproporcionado. El enfoque se centrará en el funcionamiento nominal del sistema en dos escenarios principales: en tierra y durante la fase de crucero.

3.5.2 Análisis de Sistema

Continuando con la metodología Arcadia, y tras haber completado el Análisis Operacional, el Análisis de Sistema (o Análisis de Necesidades del Sistema) se enfoca en definir **qué debe hacer el sistema** para satisfacer las **Capacidades Operacionales (OC)** identificadas. En esta etapa, se empieza a abrir la "caja negra" para definir las funciones principales del sistema y los flujos funcionales entre ellas y con los actores externos.

Delimitación del Sistema y Actores Externos

Para establecer el alcance de nuestro estudio, resulta fundamental definir con precisión los límites del sistema bajo consideración y los actores externos que interactúan con él.

Motores y/o Unidad de Potencia Auxiliar (APU): Estos componentes actúan como la fuente principal de aire de sangrado de alta presión y temperatura. La disponibilidad y las características de este aire (presión, temperatura, caudal) son parámetros críticos que influyen directamente en el rendimiento del sistema ACPC.

Sistema Eléctrico de la Aeronave: El sistema ACPC requiere energía eléctrica para el funcionamiento de componentes esenciales como ventiladores, actuadores de válvulas, sensores y unidades de control. La fiabilidad del suministro eléctrico es un requisito indispensable.

Sistema de Indicación: Es el encargado de mostrar a la tripulación el estado y los parámetros de funcionamiento del ACPC a través de paneles de control e indicadores en cabina (e.g., ECAM/EICAS), siendo crucial para la conciencia situacional.

Sistema Neumático: El ACPC forma parte integral del sistema neumático general, del cual recibe el aire de sangrado y con el que interactúa para gestionar su distribución y regulación.

Atmósfera Externa: Juega un doble papel: es la fuente de aire fresco para la refrigeración primaria de los PACKs de aire acondicionado (ram air) y, a la vez, actúa como el sumidero de calor y del aire viciado que es expulsado de la cabina a través de las válvulas de flujo (outflow valves).

Fuente Externa de Aire en Tierra: Corresponde a un equipo de apoyo en tierra que suministra aire acondicionado (PCA - Pre-conditioned Air) o aire a alta presión a la aeronave. Permite climatizar la cabina sin operar la APU o los motores, reduciendo el consumo de combustible y el ruido en la plataforma.

Cabina: Es el espacio físico (incluyendo bodegas) que el sistema debe presurizar y climatizar. Actúa como el "cliente" del sistema, conteniendo los sensores de temperatura y presión que proporcionan el feedback necesario para que los controladores del ACPC mantengan las condiciones de confort y seguridad.

Tripulación de Vuelo: Son los operadores principales del sistema. A través del panel de control, establecen los modos de funcionamiento (automático, manual) y seleccionan las temperaturas deseadas para las diferentes zonas (cabina de vuelo, cabina de pasajeros), además de gestionar cualquier contingencia.

Tripulación de Cabina/Auxiliares de Vuelo: Interactúan con el sistema principalmente para asegurar el confort de los pasajeros. En muchas aeronaves, tienen la capacidad de realizar ajustes finos en la temperatura de las diferentes zonas de la cabina y son los primeros en reportar cualquier anomalía de confort (olores, temperaturas extremas) a la tripulación de vuelo.

En la **Figura 3.16** se presenta el **Diagrama de Contexto del Sistema (CSA)**, el cual representa gráficamente el sistema ACPC en su entorno operativo, mostrando las interacciones entre el sistema y los actores externos identificados.

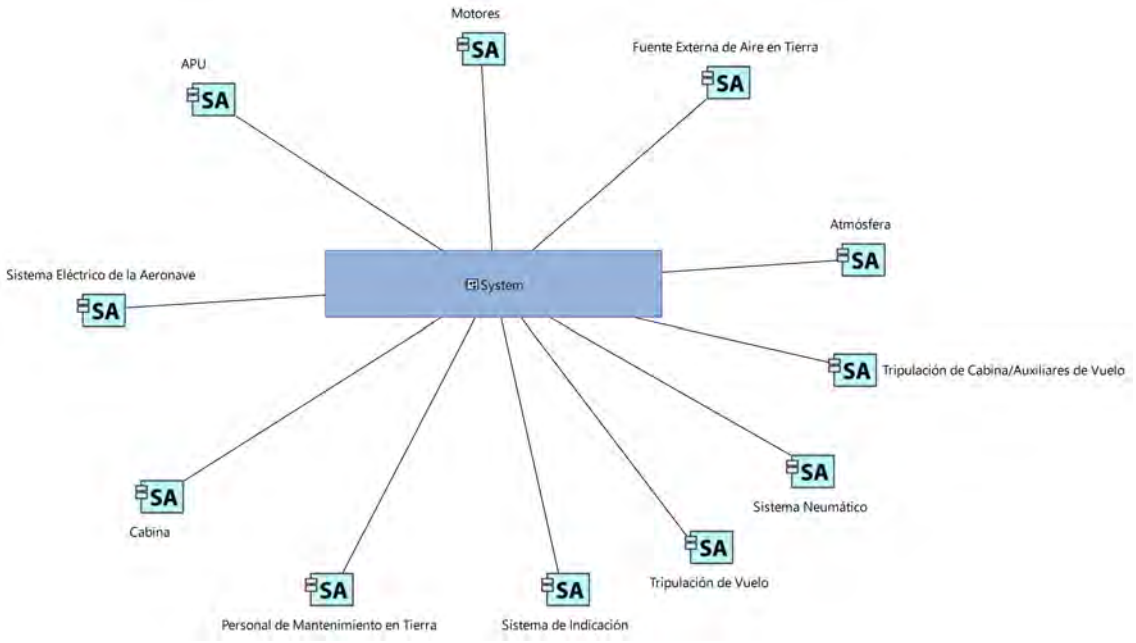

Figura 3.16 Esquema CSA.

El diagrama MCB actúa como el puente conceptual entre los requisitos de alto nivel de la misión y el inicio del desglose funcional. Cada capacidad identificada en el esquema deberá ser satisfecha por una o varias cadenas funcionales del sistema. De este modo, el MCB estructura y guía el Análisis Lógico, asegurando que cada función que se defina posteriormente tenga un propósito claro y trazable hasta una capacidad y, en última instancia, hasta una misión. Es el primer nivel de descomposición que permite pasar de los objetivos abstractos a un conjunto estructurado de habilidades que el sistema deberá implementar.

Se presenta a continuación el **Mission Capabilities Bank** (**Figura 3.17**). Aparecen todas las entidades del sistema involucradas con las capacidades correspondientes. Se ha hecho descender las capacidades de proveer presurización, acondicionamiento térmico y ventilación de una misma misión llamada **proveer condiciones adecuadas de cabina**. Las misiones vienen a ser similares a las capacidades pero refiriéndose a un nivel más alto. Además, se ha añadido una capacidad común a estas tres últimas que es la de **captar el aire externo**, importante porque es la que realizan los turbofanes para enviarlo a continuación al sistema de sangrado.

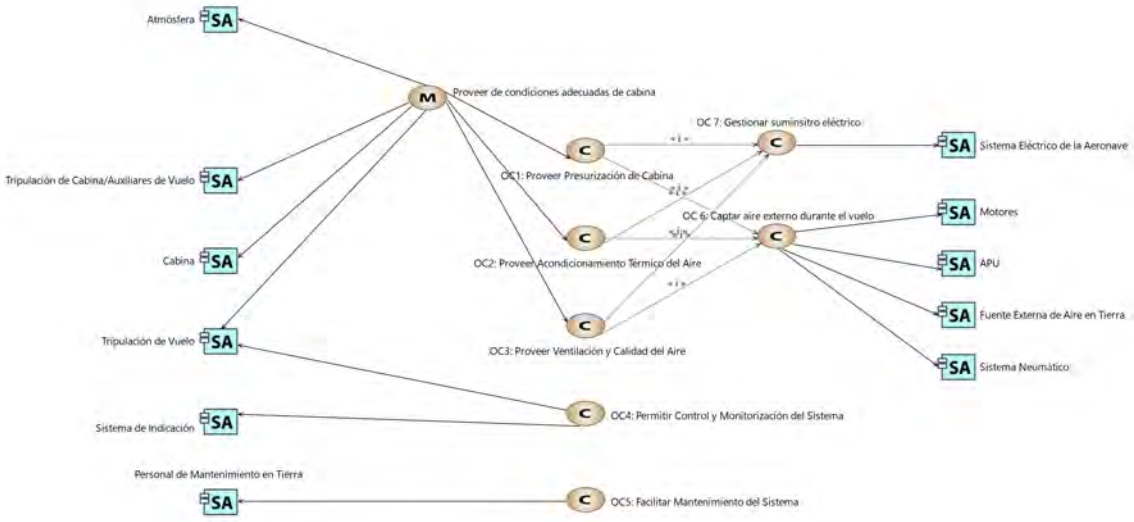

Figura 3.17 Esquema MCB.

Definición de las Funciones del Sistema (System Functions - SF)

Las funciones del sistema se derivan de las Capacidades Operacionales (OC) y describen las acciones que el sistema ACPC debe realizar. Estas funciones, interconectadas mediante intercambios funcionales[4], formarán las cadenas funcionales.

Funciones de Presurización (Relacionadas con OC1)

- **Controlar Presión de Cabina:** Regular activamente la presión dentro de la cabina modulando el flujo de aire de salida para seguir un cronograma de presurización y mantener la presión diferencial (ΔP) dentro de límites.
- **Medir Parámetros de Presión:** Determinar continuamente la altitud de cabina, la ΔP y la tasa de cambio de presión.
- **Proteger Contra Extremos de Presión:** Implementar mecanismos para prevenir sobrepresión o ΔP negativa excesiva.
- **Permitir Despresurización de Emergencia:** Proveer la capacidad de despresurizar la cabina de forma controlada (manual o automática).

Figura 3.18 Esquema SFCD: Funciones de Presurización.

Funciones de Acondicionamiento Térmico (Relacionadas con OC2)

- **Captar y Regular Aire de Origen.**
- **Enfriar Aire de Origen:** Reducir la temperatura del aire de sangrado mediante intercambiadores de calor y una máquina de ciclo de aire (ACM).
- **Calentar Aire:** Aumentar la temperatura del aire suministrado mezclando aire caliente (bypass del PACK) con aire frío, o mediante un sistema "trim air".

[4] En los diagramas se utilizara "TUBX" para referirnos a la interacción entre funciones que supone el transforte del fluido.

- **Controlar Temperatura de Suministro por Zona:** Regular la temperatura del aire entregado a las diferentes zonas según las selecciones de la tripulación.
- **Medir Temperaturas Clave:** Medir temperaturas en puntos críticos del sistema y en las zonas acondicionadas.

Figura 3.19 Esquema SFCD: Funciones de Acondicionamiento Térmico.

Funciones de Ventilación y Calidad del Aire (Relacionadas con OC3)

- **Distribuir Aire Acondicionado:** Canalizar el aire tratado a las diferentes zonas.
- **Filtrar Aire:** Eliminar partículas y contaminantes del aire de suministro y/o recirculado.
- **Controlar Tasa de Ventilación:** Asegurar un flujo de aire fresco y una tasa de renovación adecuados.
- **Gestionar Recirculación de Aire:** Controlar la mezcla de aire fresco y recirculado para optimizar la eficiencia.
- **Prevenir Contaminación del Aire:** Implementar medidas para evitar la introducción de contaminantes.

Figura 3.20 Esquema SFCD: Funciones de ventilación.

Funciones de Control y Monitorización (Relacionadas con OC4)

- **Estudio de Condiciones:** Analizar los requerimientos de aire necesarios para la operación de la aeronave.
- **Monitorización en Vuelo:** Supervisar continuamente las condiciones del sistema durante las distintas fases de vuelo.
- **Configuración del Sistema ACOC:** Seleccionar el modo de operación automático (AUTO) y verificar que la presurización está programada correctamente.
- **Ajuste Dinámico de Condiciones:** Modificar los parámetros del sistema si es solicitado por la tripulación de cabina o si las condiciones de vuelo cambian.
- **Envío de Controles:** Transmitir las órdenes y comandos del piloto a los actuadores del sistema.
- **Gestión del Aire de Sangrado:** Administrar y controlar el suministro de aire de sangrado proveniente de los motores.

Figura 3.21 Esquema SFCD: Funciones de control y monitorización.

Funciones de Mantenimiento (Relacionadas con OC5)

(Se ha decidido no continuar con esta función por exceder el alcance del proyecto)

- **Realizar Auto-Diagnóstico:** Ejecutar pruebas internas para detectar fallos.
- **Registrar y Comunicar Datos de Fallo:** Almacenar y permitir el acceso a la información de fallos.
- **Soportar Pruebas en Tierra:** Permitir la activación y prueba de funciones por el personal de mantenimiento.

Funciones de Aire de Sangrado (Relacionadas con OC6)

- **Ruta 1 (Suministro en Tierra):** El sistema toma aire de una **fuente externa**, lo procesa a través del **suministro de aire en tierra** y luego lo dirige mediante la **canalización y redireccionamiento** hacia la etapa de gestión final.
- **Ruta 2 (Conversión a Sangrado):** De forma alternativa, el aire de la **fuente externa** es sometido a un proceso de **conversión a aire de sangrado**, para después ser conducido por la **canalización y redireccionamiento** hacia la gestión del suministro.
- **Ruta 3 (Recolección Externa):** El sistema realiza una **recolección de aire externo** (proceso que puede ocurrir tanto en tierra como en vuelo) y lo envía a través de la **canalización y redireccionamiento** para su posterior gestión.

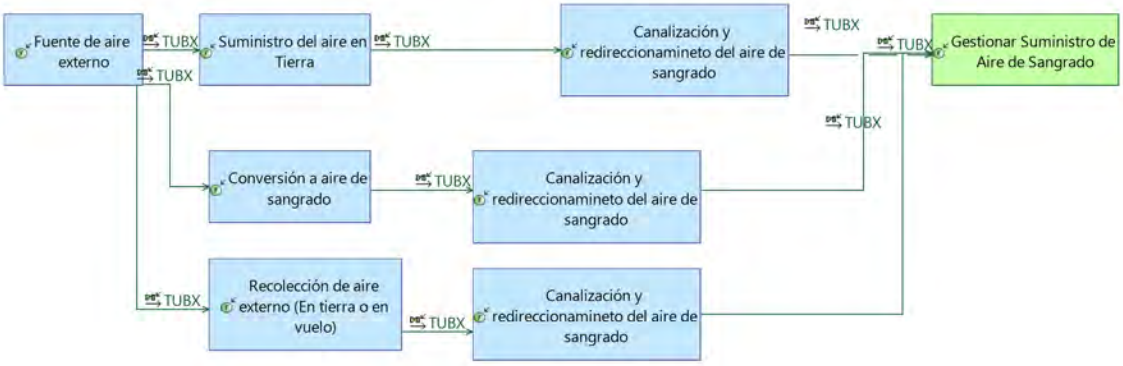

Figura 3.22 Esquema SFCD: Funciones de aire de sangrado.

Funciones de Suministro Eléctrico (Relacionadas con OC7)

- **Gestionar Suministro Eléctrico:** Recibir y gestionar la energía eléctrica para la operación de los componentes del ACPC.

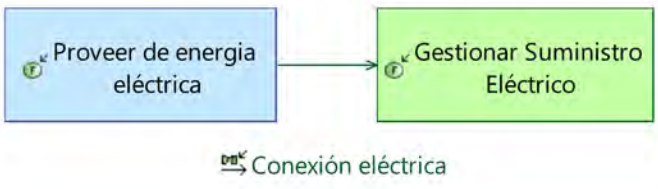

Figura 3.23 Esquema SFCD: Funciones de Suministro Eléctrico.

Diagrama de Arquitectura del Sistema (SAB). Representado en Figura 3.24

En el SAB (System Architecture Blank) generalmente se encuentra toda la información: funciones, entidades y cadenas funcionales que se ha descrito para nuestro sistema. Se representan las distintas

cadenas funcionales con un color distinto e interconectadas entre sí. El mapeado de cada función en su entidad permite saber quién las realiza y, a su vez, se pueden apreciar las interacciones entre entidades y sistemas.

3.5.3 Análisis Lógico

El **Análisis Lógico** define la solución conceptual del sistema, describiendo **cómo** funcionará para cumplir con los requisitos de una forma abstracta y tecnológicamente neutra. En esta fase, se transita de una visión de "caja negra" a una de "caja de cristal", revelando la organización funcional interna sin tomar decisiones sobre los componentes tecnológicos o físicos que se emplearán en la implementación.

El objetivo es establecer una arquitectura lógica coherente a través de las siguientes actividades clave:

- **Descomposición Funcional y Agrupación:** Se descomponen las funciones de alto nivel en subfunciones más simples. Estas se agrupan de manera lógica para definir los **componentes lógicos**, que son bloques funcionales abstractos.
- **Asignación de Funciones (LFCD):** Cada función y subfunción es asignada a un componente lógico específico, asegurando que toda la funcionalidad requerida está cubierta.
- **Definición de Interfaces:**[5] Se identifican y definen los flujos de información, energía o materia que conectan los diferentes componentes lógicos entre sí.

La profundidad de este análisis depende del contexto del diseño. Por ejemplo, un subsistema que será adquirido de un proveedor externo puede tratarse como un único componente lógico de "caja negra", analizando únicamente sus interfaces externas sin necesidad de detallar su estructura interna.

Los principales artefactos (entregables) de esta fase son:

- **Diagrama de Descomposición de Componentes Lógicos (LCBD):** Es la estructura jerárquica que desglosa el sistema en subsistemas, componentes y subcomponentes lógicos.
- **Cadenas Funcionales Detalladas:** Se refina el análisis de las cadenas funcionales, buscando que cada componente lógico participe en al menos una de ellas. Esto garantiza que todos los componentes tienen un propósito definido dentro del sistema.
- **Diagrama de Arquitectura Lógica (LAB):** Es el esquema que integra visualmente los componentes del LCBD y las cadenas funcionales. Este diagrama ofrece una visión completa de la arquitectura lógica: qué componentes existen y cómo interactúan para ejecutar las funciones del sistema.

Definición de las Funciones del Sistema (System Functions - SF)

Sucede algo similar a lo que ya hicimos en la capa de análisis de sistema.A partir de las capacidades operacionales, el siguiente paso consiste en descomponerlas en un conjunto de funciones de sistema (System Functions). Estas funciones describen las acciones concretas que el sistema debe ejecutar. A continuación, se detallan y agrupan estas funciones, formando las cadenas funcionales que posteriormente serán asignadas a los componentes lógicos de la arquitectura.

Cadena Funcional 1: Control de la Presión

- **Medir Parámetros de Presión:** Mide la presión diferencial y absoluta para informar al controlador de presurización.

[5] Nosotros este punto no lo llegaremos a desarrollar por salirse del alcance del proyecto.

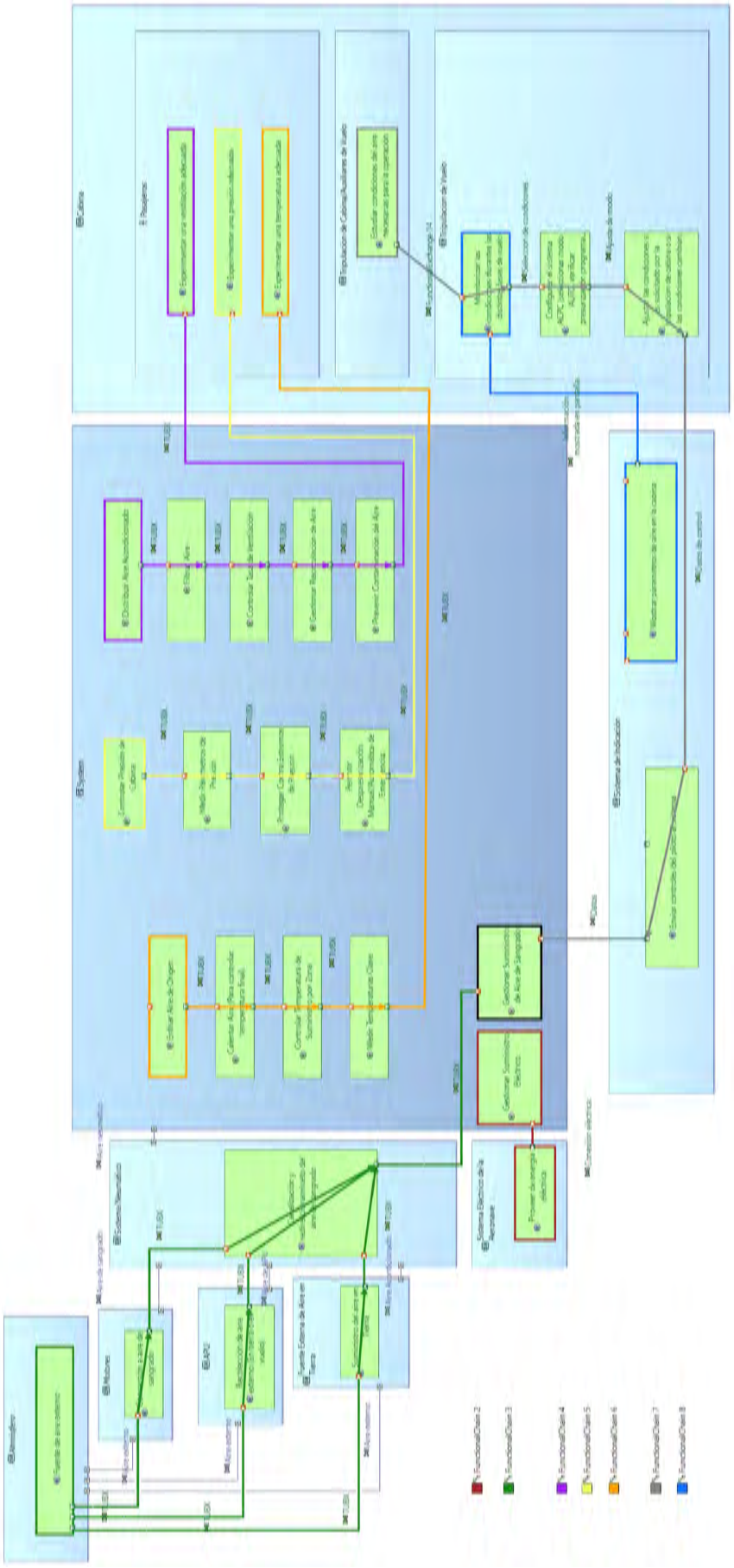

Figura 3.24 Esquema SAB.

- **Controla modo de actuación de la válvula de salida:** Gestiona si la Outflow Valve opera en modo automático, manual o de emergencia.
- **Apertura paso de aire Válvula de Salida:** Corresponde a la acción física de la válvula al abrirse o cerrarse para regular la presión.

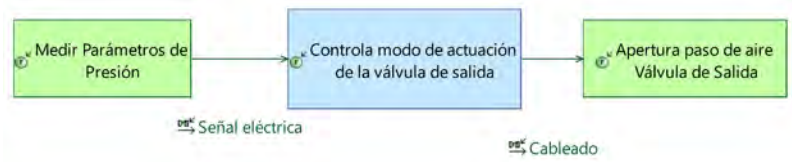

Figura 3.25 Esquema LFCD:Control de la Presión.

Cadena Funcional 2: Recirculación y Mezcla de Aire

- **Absorber aire de la cabina:** Extrae un porcentaje del aire de la cabina para ser filtrado y reutilizado.
- **Filtrar Aire:** Elimina partículas y contaminantes del aire recirculado mediante filtros de alta eficiencia (HEPA).
- **Mezcla de aire acondicionado, de sangrado, recirculado:** Combina aire fresco (acondicionado) con aire recirculado y aire de trimado caliente.
- **Distribuir Aire Acondicionado:** Impulsa la mezcla de aire final hacia el sistema de distribución de la cabina.

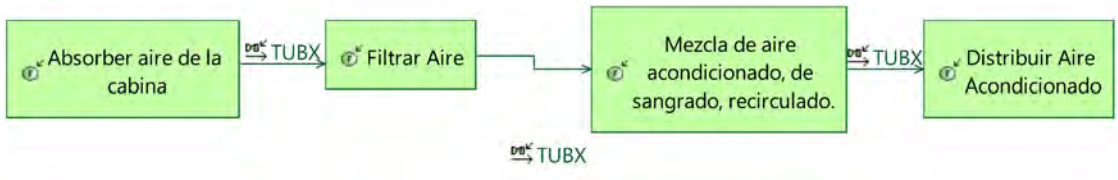

Figura 3.26 Esquema LFCD:Recirculación y Ventilación del Aire.

Cadena Funcional 3: Ciclo de Aire Acondicionado

- **Regular apertura Válvula DCTV:** Modula la válvula de control de flujo que admite el aire de sangrado en el PACK.
- **Enfriar aire de sangrado:** Reduce la temperatura inicial del aire de sangrado en el primer intercambiador de calor.
- **Comprimir y aumentar temperatura:** El compresor de la ACM aumenta la presión y temperatura del aire para mayor eficiencia.
- **Enfriar aire de ACM:** Vuelve a enfriar el aire, ya comprimido, en el segundo intercambiador de calor.
- **Expandir y reducir temperatura del aire:** La turbina de la ACM expande el aire, provocando la drástica caída de temperatura final.
- **Regular apertura Válvula DCTV:** Controla una válvula de control de temperatura para regular la salida del PACK.

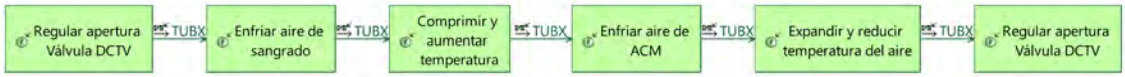

Figura 3.27 Esquema LFCD:Acondicionamiento del Aire.

Cadena Funcional 4: Control de Temperatura por Zonas

- **Regular apertura Válvula DCTV / de válvulas por zona:** Modula las válvulas de trimado que añaden aire caliente para ajustar la temperatura de una zona específica.
- **Medir Temperaturas Clave:** Monitoriza la temperatura en los conductos para proporcionar feedback al controlador.
- **Controlar temperatura de cada parte del avión:** Procesa las solicitudes y mediciones para comandar las válvulas de trimado de cada zona.

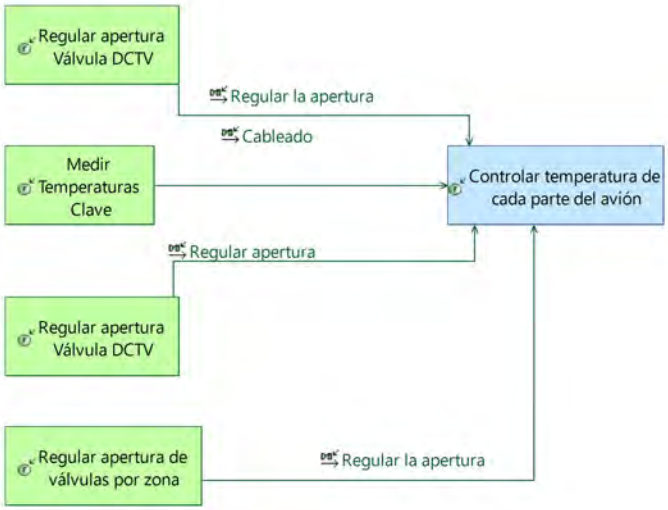

Figura 3.28 Esquema LFCD:Control de la temperatura por zonas.

Cadena Funcional 5: Gestión del Aire de Sangrado

- **Entregar aire de presión intermedia / alta:** Extrae aire de las etapas correspondientes del compresor del motor.
- **Controlar el suministro de aire de sangrado:** Regula la presión y el flujo del aire extraído para proteger el sistema.
- **Control del aire de presión intermedia / alta:** Selecciona automáticamente la etapa del motor (IP o HP) para optimizar el rendimiento.
- **Suministrar aire externo del fan / Control del aire del fan:** Utiliza aire del fan del motor para tareas de baja presión como la ventilación en tierra.
- **Regular suministro de aire de la APU:** Controla el aire suministrado por la Unidad de Potencia Auxiliar (APU) en tierra.
- **Corte en caso de fallo de motor:** Cierra automáticamente la válvula de sangrado de un motor si este falla.
- **Prevenir Contaminación del Aire:** Detecta contaminantes en el aire de sangrado y aísla la fuente si es necesario.

- **Redirigir aire a sistemas externos:** Deriva aire de sangrado a otros sistemas como el anti-hielo de las alas.
- **Controla flujo de aire que entra en el PACK:** Ajusta la cantidad de aire que se envía a los PACKs de aire acondicionado.
- **Suministrar aire acondicionado en tierra:** Permite la conexión a una unidad de aire acondicionado externa en el aeropuerto.
- **Distribuir Aire Acondicionado:** Envía el aire de la fuente (motor, APU o tierra) al sistema de distribución.

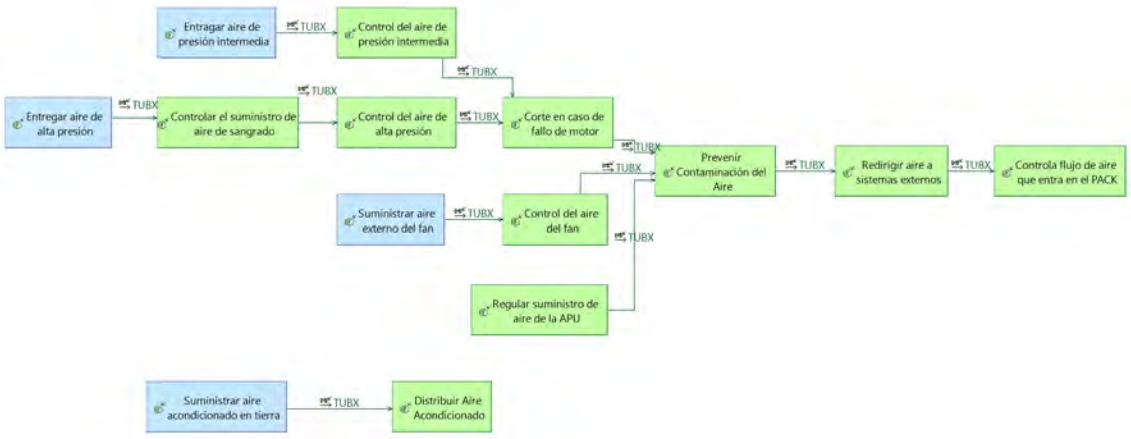

Figura 3.29 Esquema LFCD:Gestión del Aire de Sangrado.

Cadena Funcional 6: Monitorización e Interfaz con la Tripulación

- **Adquirir Datos de Sistemas Externos de la Aeronave:** Recibe información de altitud, velocidad y fase de vuelo de otros sistemas.
- **Proveer Indicaciones y Alertas a la Tripulación:** Genera los mensajes y alertas que se muestran a los pilotos en el EICAS/ECAM.
- **Mostrar parámetros de aire en la cabina:** Presenta los datos clave del sistema (presión, temperatura, estado de válvulas) en las pantallas.
- **Monitorizar las condiciones durante las distintas fases de vuelo:** Adapta el comportamiento del sistema según si el avión está en tierra, ascenso, crucero o descenso.

Figura 3.30 Esquema LFCD:Monitorización e Interfaz con la tripulación.

Cadena Funcional 7: Suministro Eléctrico (Representada en Figura 3.31)

- **Proveer de energía eléctrica:** La aeronave suministra la potencia eléctrica requerida por el sistema.
- **Gestionar Suministro Eléctrico:** El controlador del sistema distribuye la energía a sus sensores, actuadores y ventiladores.

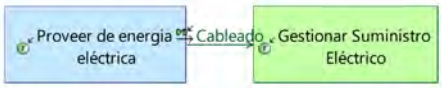

Figura 3.31 Esquema LFCD: Gestión del siministro eléctrico.

Cadena Funcional 8: Circuito de Aire de Impacto (Ram Air)

- **Proveer de aire externo / Extraer aire dinámico exterior:** Captura el flujo de aire exterior a través de una toma de aire de impacto.
- **Calentar aire de ram primario / secundario:** El aire de impacto absorbe calor al pasar por los intercambiadores de calor del PACK.
- **Absorbe aire de impacto para calentado en ACM:** Utiliza el aire de impacto como sumidero térmico para enfriar el aire del ciclo de sangrado.
- **Proveer de aire externo (salida):** Expulsa el aire de impacto, ya caliente, de vuelta a la atmósfera.

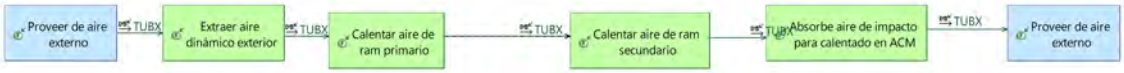

Figura 3.32 Esquema LFCD:Circuito Aire de Impacto (Ram Air).

Cadena Funcional 9: Mezcla y Distribución Final

- **Regular apertura Válvula DCTV:** Modula una válvula de control de temperatura para la mezcla general.
- **Redirigir aire a sistemas externos:** Desvía aire a otros consumidores antes de la distribución en cabina.
- **Permite conectar sistema de sangrado con cámara de mezcla directamente:** Activa una ruta de bypass para escenarios no estándar.
- **Mezcla de aire acondicionado, de sangrado, recirculado:** Combina todas las fuentes de aire en una cámara de mezcla (mixing plenum).
- **Distribuir Aire Acondicionado:** Envía la mezcla de aire hacia las válvulas de zona y la Outflow Valve.
- **Regular apertura de válvulas por zona:** Realiza el ajuste fino de la temperatura para cada zona de la cabina.
- **Apertura paso de aire Válvula de Salida:** Controla la evacuación de aire de la cabina para la presurización.
- **Distribuir aire por todos los lugares de la cabina:** Dirige el aire desde el plenum hacia las salidas individuales de aire en la cabina.

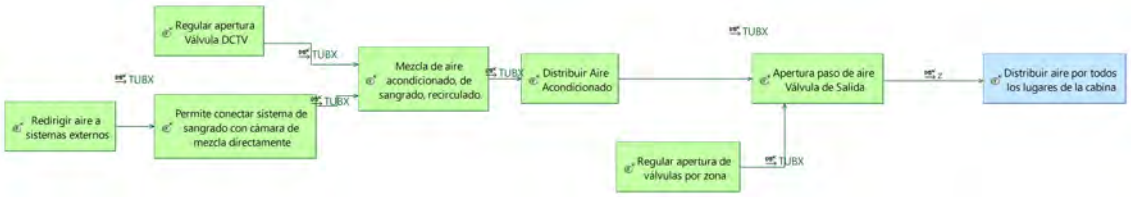

Figura 3.33 Esquema LFCD: Mezcla y distribución final.

LCBD: Descomposición de Componentes. Representado en Figura 3.34

Sistema de Aire Acondicionado y Presurización

- **1.0 Subsistema de Acondicionamiento**
  - **1.1 Válvula de Control de Flujo (Flow Control Valve - FCV):** Regula el caudal de aire de sangrado que ingresa al PACK.
  - **1.2 Máquina de Ciclo de Aire (Air Cycle Machine - ACM):** El corazón del PACK, donde se enfría el aire.
    - **1.2.1 Sección del Compresor:** Comprime el aire para aumentar su temperatura antes del segundo intercambiador.
    - **1.2.2 Sección de la Turbina:** Expande el aire, provocando una drástica caída de su temperatura.
    - **1.2.3 Eje común (Shaft):** Une mecánicamente el compresor y la turbina.
  - **1.3 Intercambiadores de Calor (Heat Exchangers):** Enfrían el aire de sangrado usando el aire de impacto (ram air).
    - **1.3.1 Intercambiador de Calor Primario.**
    - **1.3.2 Intercambiador de Calor Secundario.**
  - **1.4 Circuito de Aire de Impacto (Ram Air Circuit):** Proporciona el flujo de aire exterior para enfriar los intercambiadores.
    - **1.4.1 Toma de Aire de Impacto (Inlet).**
    - **1.4.2 Compuerta de Salida (Outlet Door).**
- **2.0 Subsistema de Control de Temperatura**
  - **2.1 Válvulas de Mezcla:** Añaden una pequeña cantidad de aire caliente al aire frío para regular la temperatura de cada zona.
  - **2.2 Controlador de Zona:** El cerebro que comanda las válvulas de mezcla según la temperatura seleccionada y medida.
  - **2.3 Sensores de Temperatura:** Miden la temperatura en los conductos y en las diferentes zonas de la cabina.
- **3.0 Subsistema de Distribución de Aire**
  - **3.1 Cámara de Mezcla:** Mezcla el aire proveniente de los PACKs y de la recirculación.
  - **3.2 Ventiladores de Recirculación:** Reintroducen una parte del aire de la cabina en el sistema para mejorar la eficiencia.
  - **3.3 Filtros de Aire:** Purifican el aire recirculado.
  - **3.4 Red de Conductos:** Transporta el aire acondicionado a todas las partes del fuselaje.

  - **3.5 Salidas de Aire Individuales (Gaspers) y Generales.**
- **4.0 Subsistema de Control de Presurización**
  - **4.1 Controladores de Presión de Cabina (Cabin Pressure Controllers - CPCs):** Usualmente dos (primario y secundario) que calculan la apertura necesaria de la válvula de salida.
  - **4.2 Válvula de Salida (Outflow Valve):** El actuador principal que regula la evacuación de aire de la cabina para controlar la presión.
  - **4.3 Válvulas de Seguridad** Protegen el fuselaje contra sobrepresión o presión negativa.
  - **4.4 Panel de Control en Cabina:** Interfaz para que la tripulación seleccione el modo de operación y la altitud de aterrizaje.

Esquema LAB de todos los componentes sin las funciones mapeadas

Las **Figuras 3.35, 3.36** y **3.37** presentan de forma agregada el **Diagrama de Arquitectura Lógica (LAB)** del sistema. Dada su complejidad, se recomienda al lector seguir el flujo principal del aire: desde los componentes de sangrado del motor y **APU** (**Figura 3.35**, izquierda), pasando por las unidades de acondicionamiento o **PACKs** (**Figura 3.36**, centro), hasta la unidad de mezcla y distribución y la propia cabina (**Figura 3.37**).

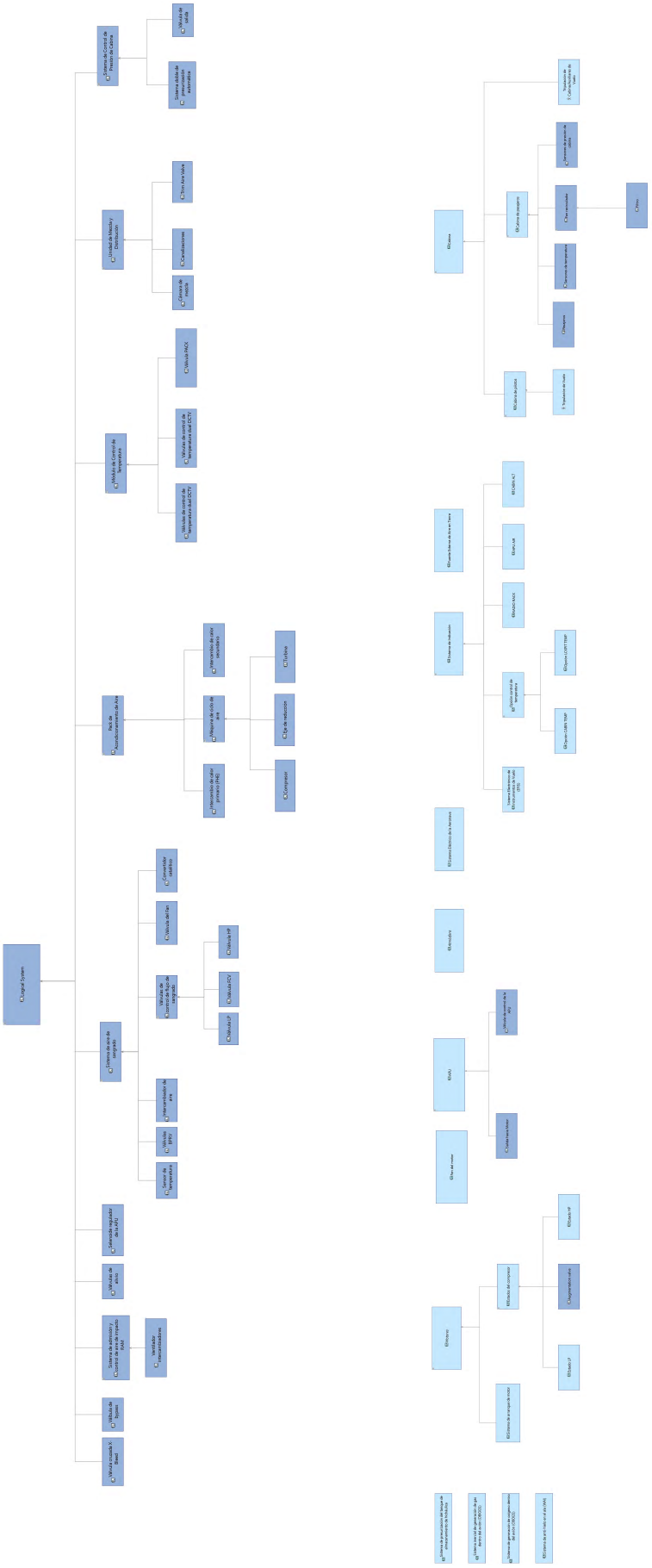

Figura 3.34 Esquema LCBD, descomposición de componentes.

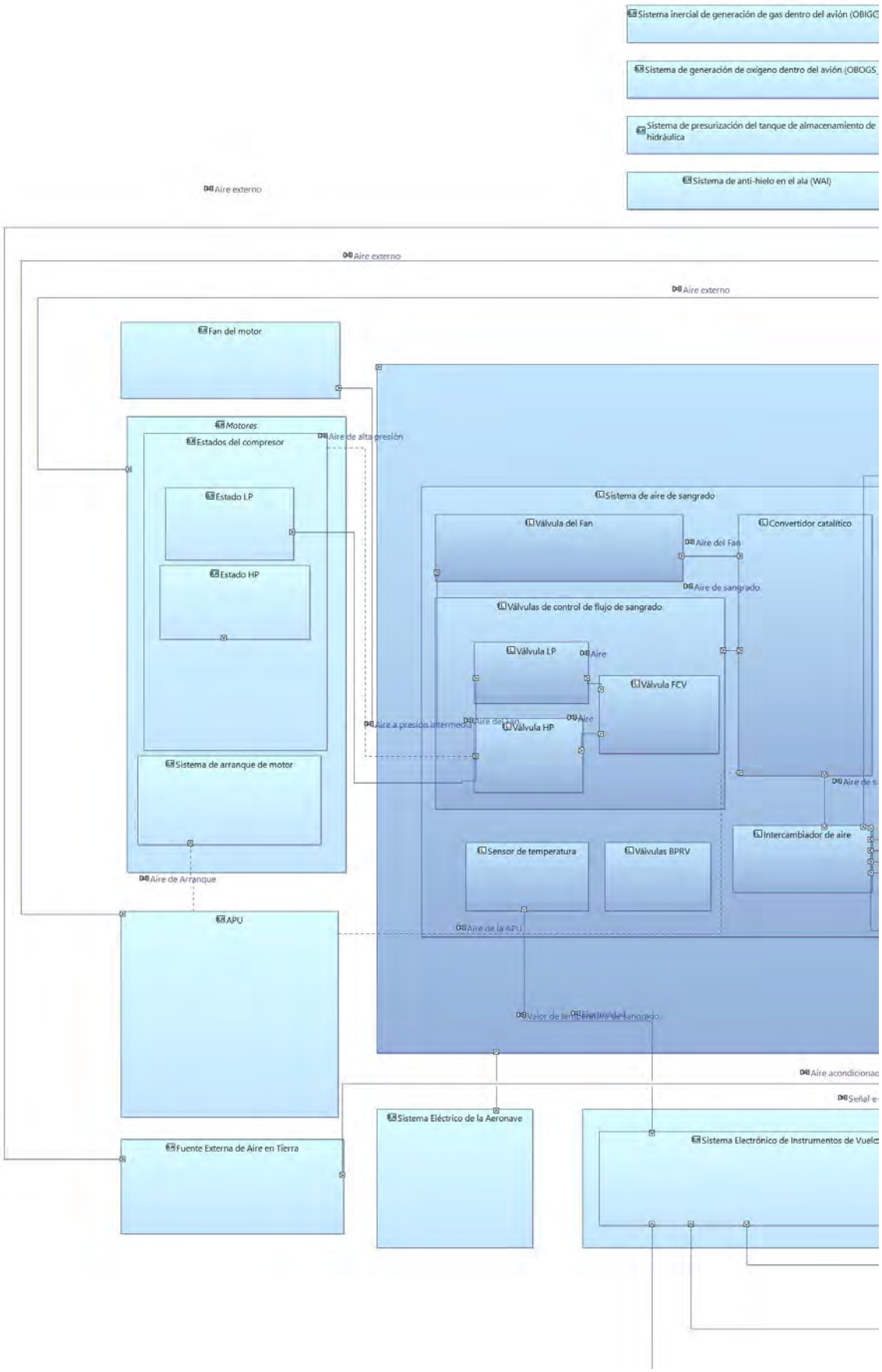

Figura 3.35 Primera parte del Esquema LAB completo.

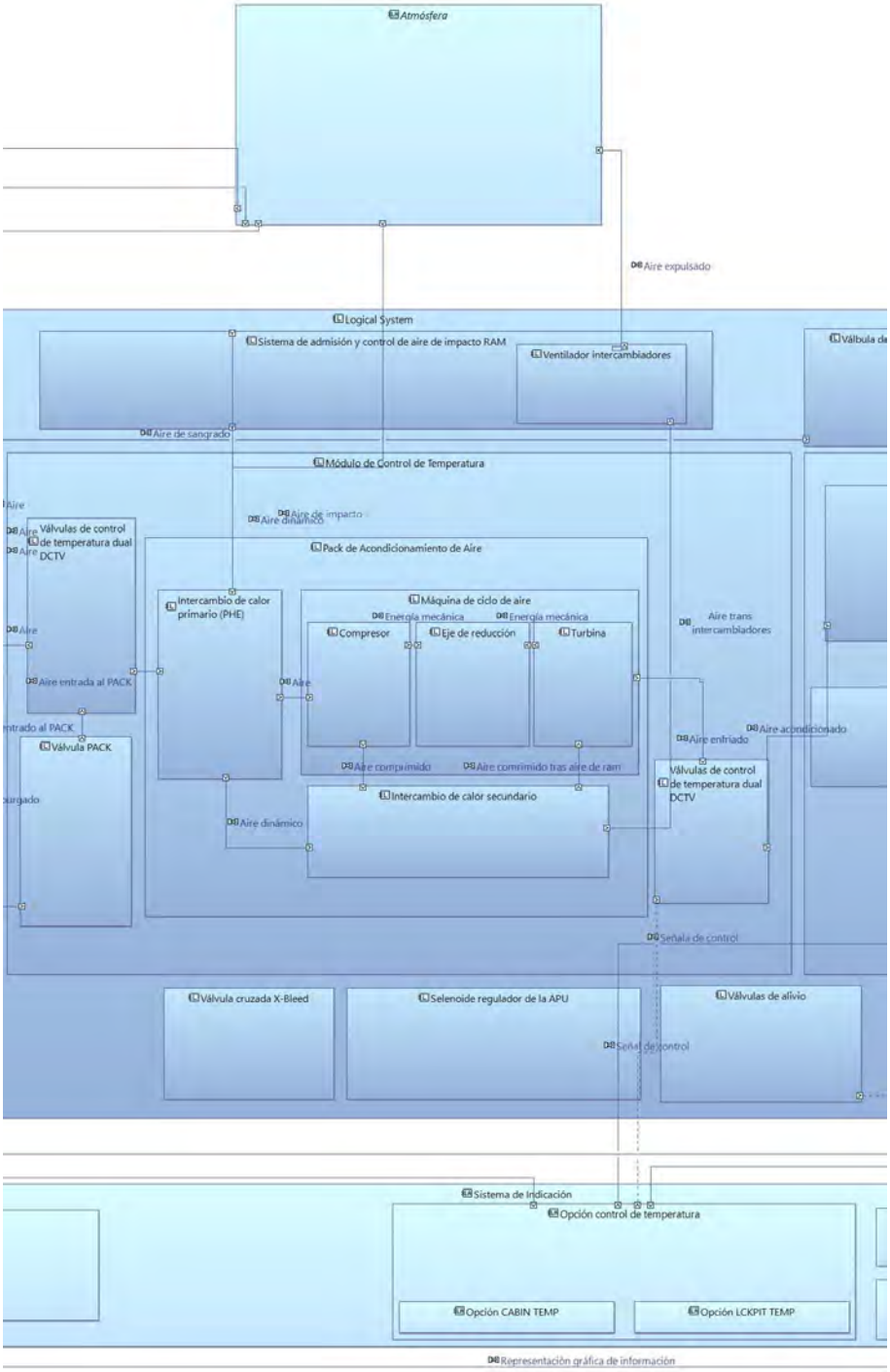

Figura 3.36 Segunda parte del Esquema LAB completo.

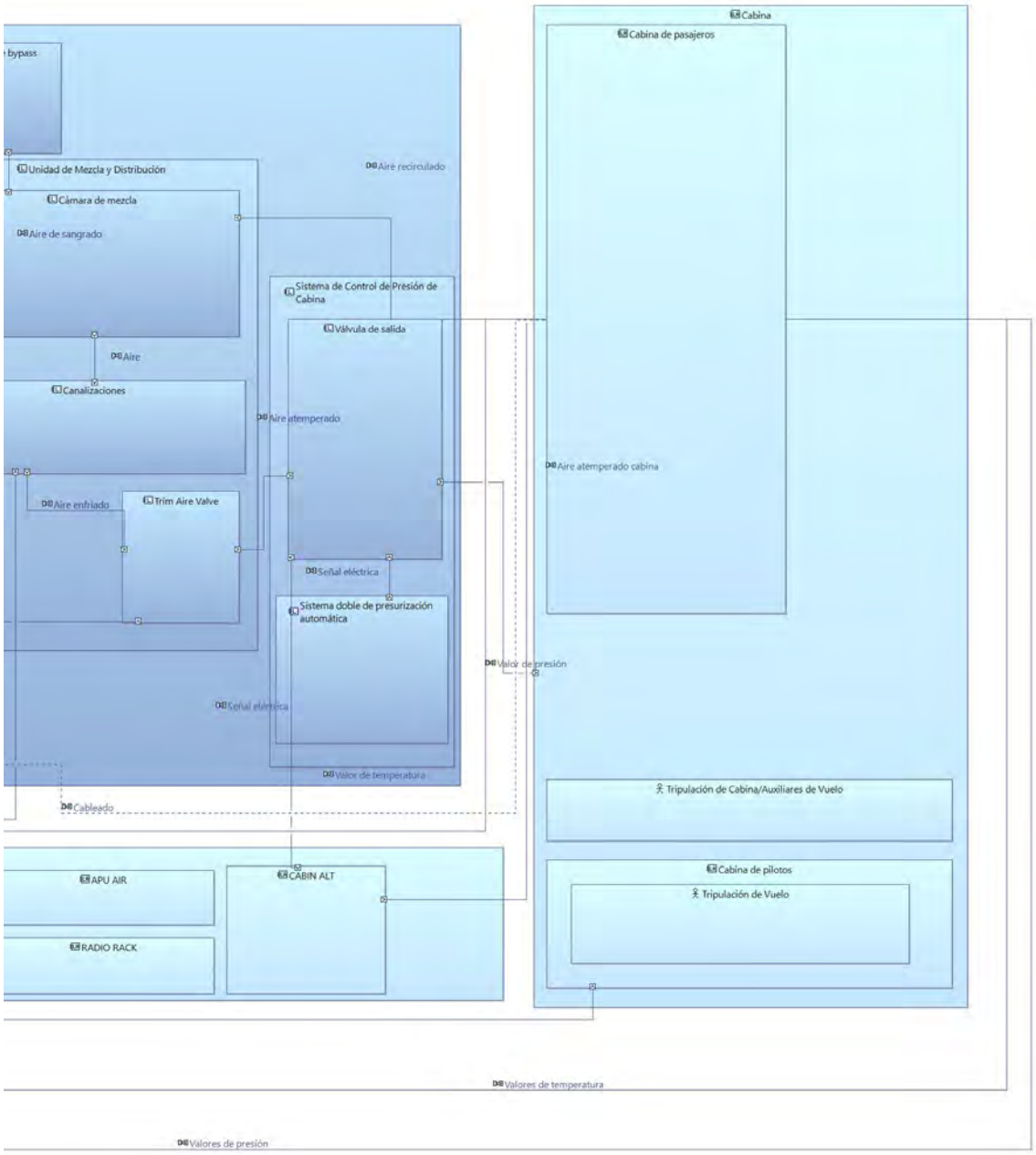

Figura 3.37 Tercera parte del Esquema LAB completo.

En los siguientes esquemas se mapearan las funciones en cada componente junto a sus cadenas funcionales correspondientes y se irán enlazando los requisitos a su componente asociado. Los requisitos son los descritos anteriormente. En los requisitos se han metido su ID y un texto con la descripción del mismo. Capella tiene muchas funcionalidades más para cumplir con los requisitos como añadir atributos en los mismos, pero no es motivo de estudio en este TFG. Por desgracia no se han pasado todos los requisitos de la normativa por lo que puede ser que en algún diagrama falte alguno. Los requisitos que son comunes a todos los subsistemas o genéricos del sistema mismo, han quedado asociados a la caja general "Logical System". Lo que interesa a grandes rasgos es ver cómo se añaden en Capella, cómo se manejan, cómo influyen en la validación y verificación del modelo y en consecuencia, cómo de correcta es la arquitectura que se ha planteado.

Esquemas LAB generados a nivel de subsistema:

Aquí se muestran los esquemas LAB asociados a cada subsistema. Se mantiene los subsistemas con los que trabaja de manera más próxima para mostrar sus interacciones. Además se añaden los requisitos específicos de cada uno.

- **Sistema de Sangrado**

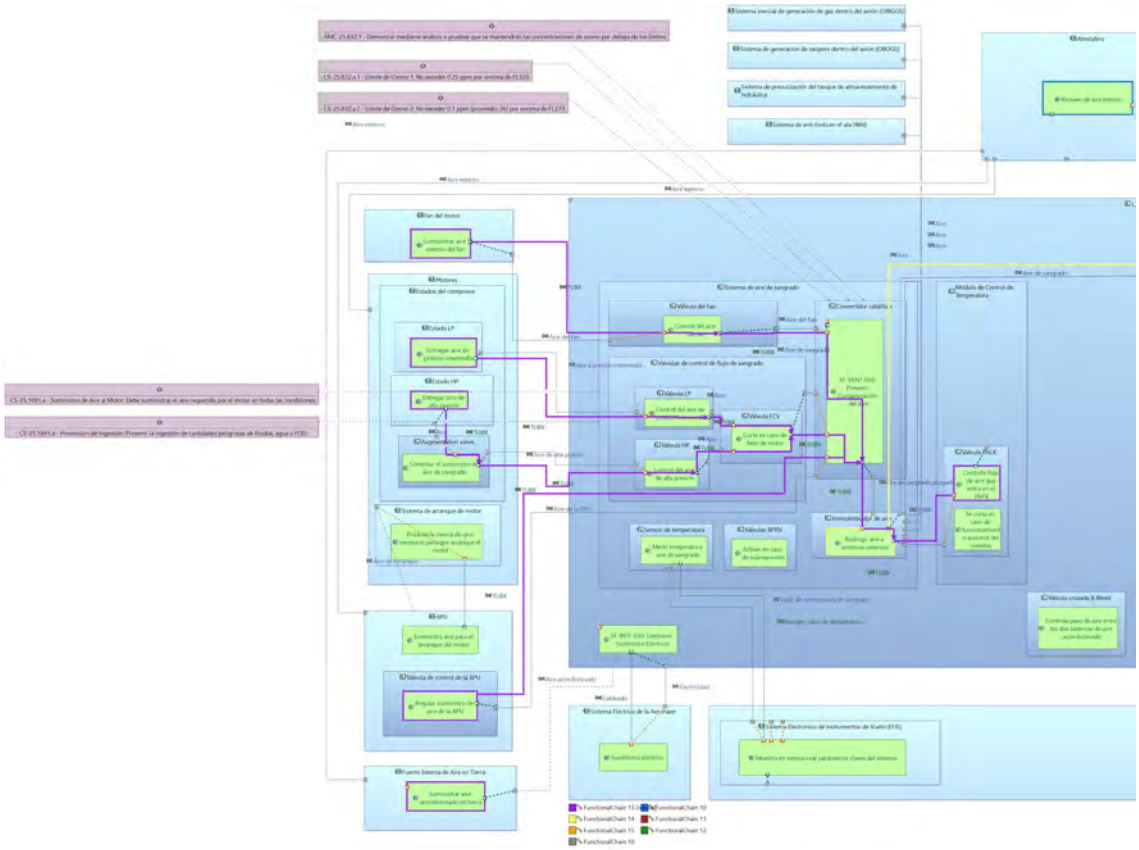

Figura 3.38 Esquema LAB: Sistema de Sangrado.

- **Sistema de control de presión de cabina**

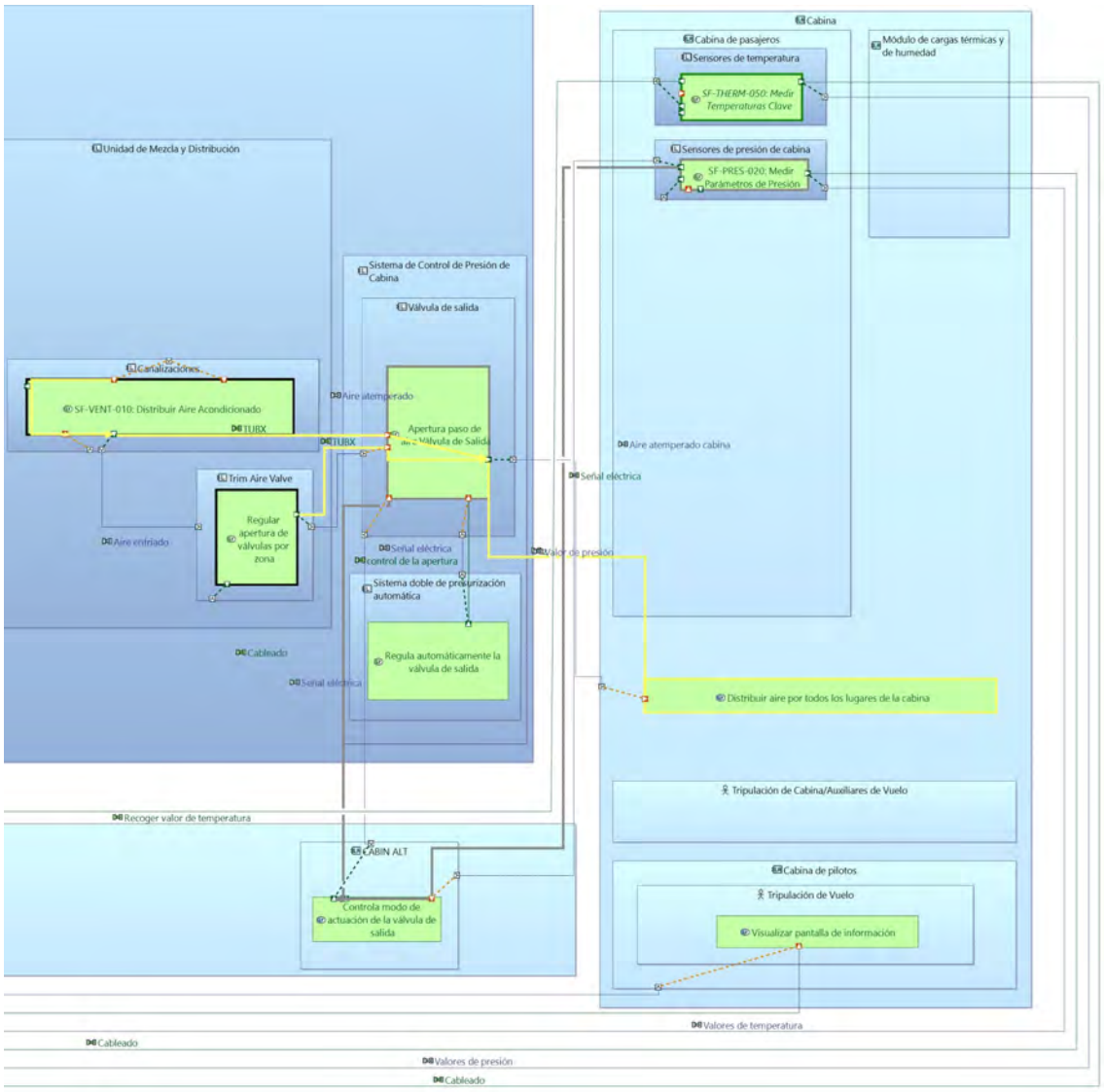

Figura 3.39 Esquema LAB: Sistema de control de presión de cabina.

- **Sistema de control del aire de impacto**

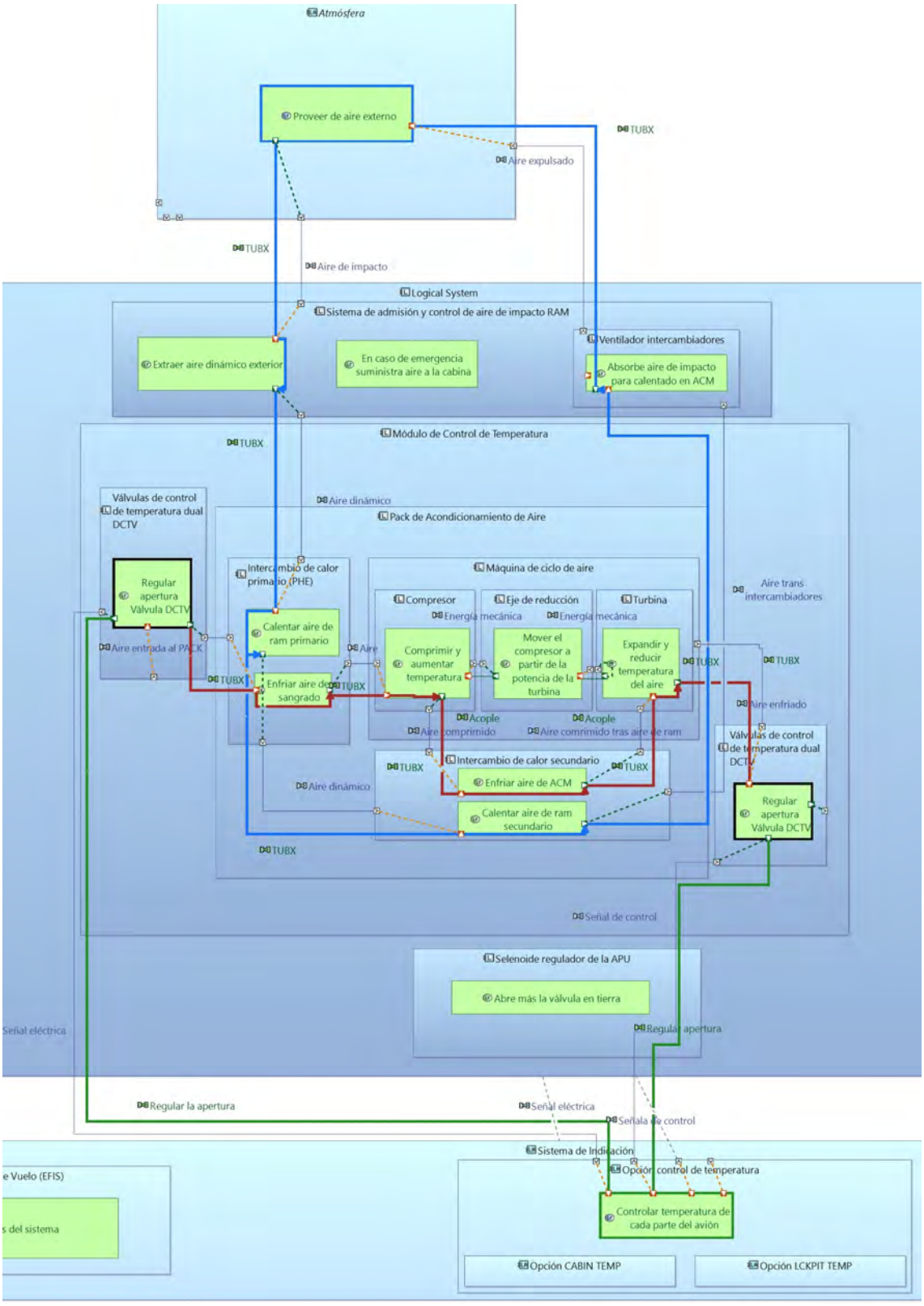

Figura 3.40 Esquema LAB: Sistema de control de aire de impacto.

- **PACK de acondicionamiento de aire**

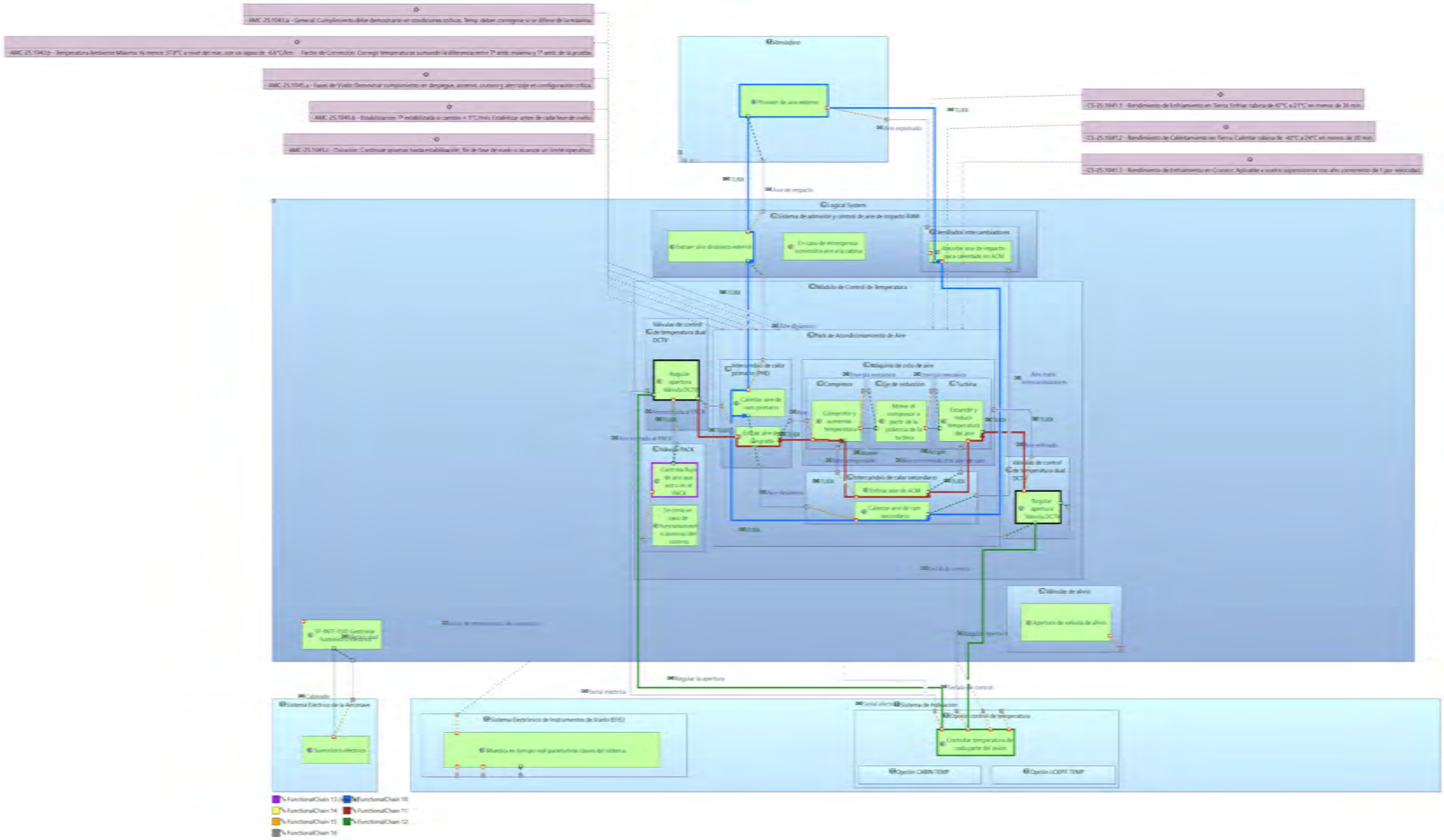

Figura 3.41 Esquema LAB: PACK de acondicionamiento de aire.

- **Unidad de mezcla y distribución**

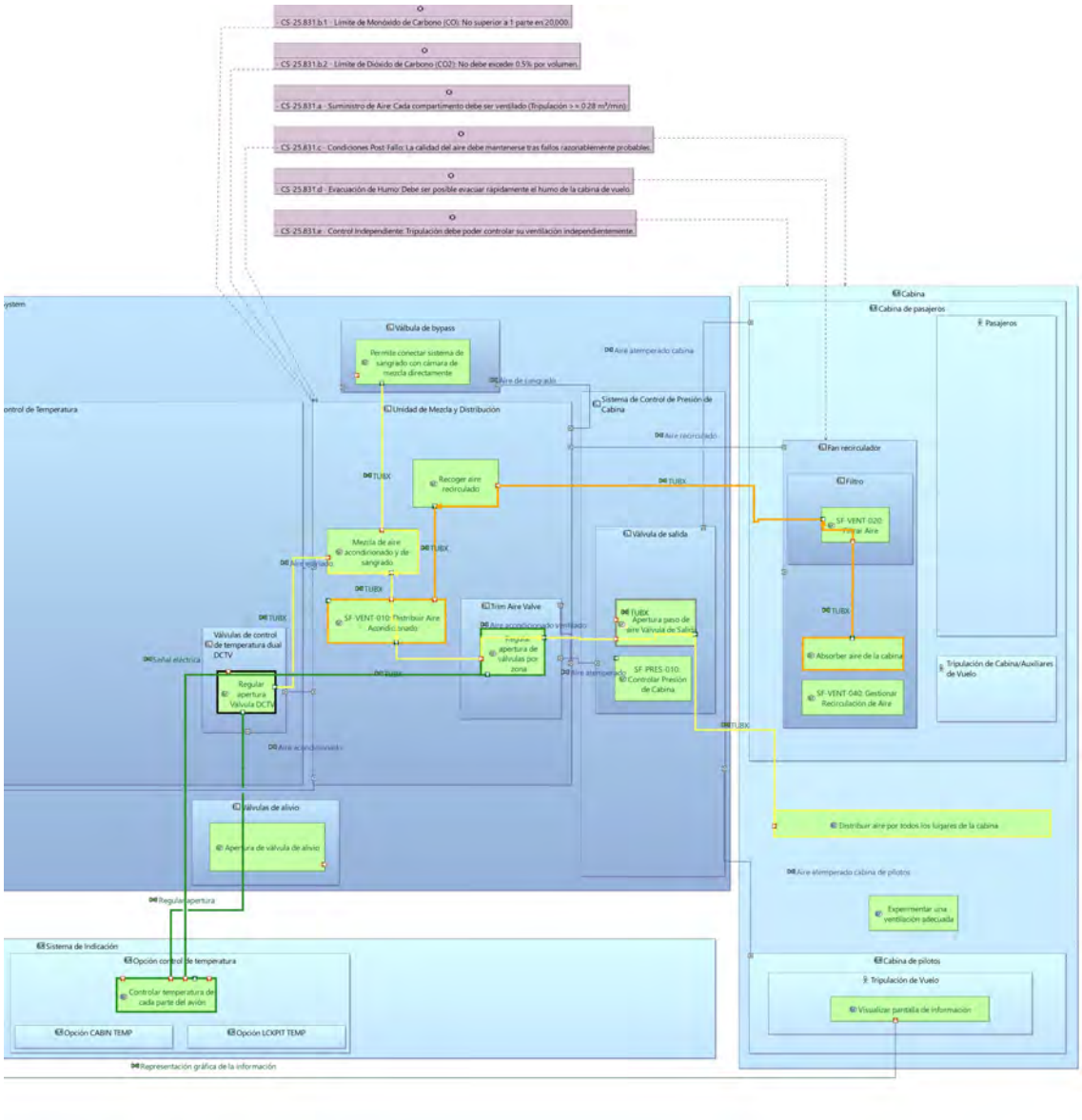

Figura 3.42 Esquema LAB: Unidad de mezcla y distribución.

- **Cabina**

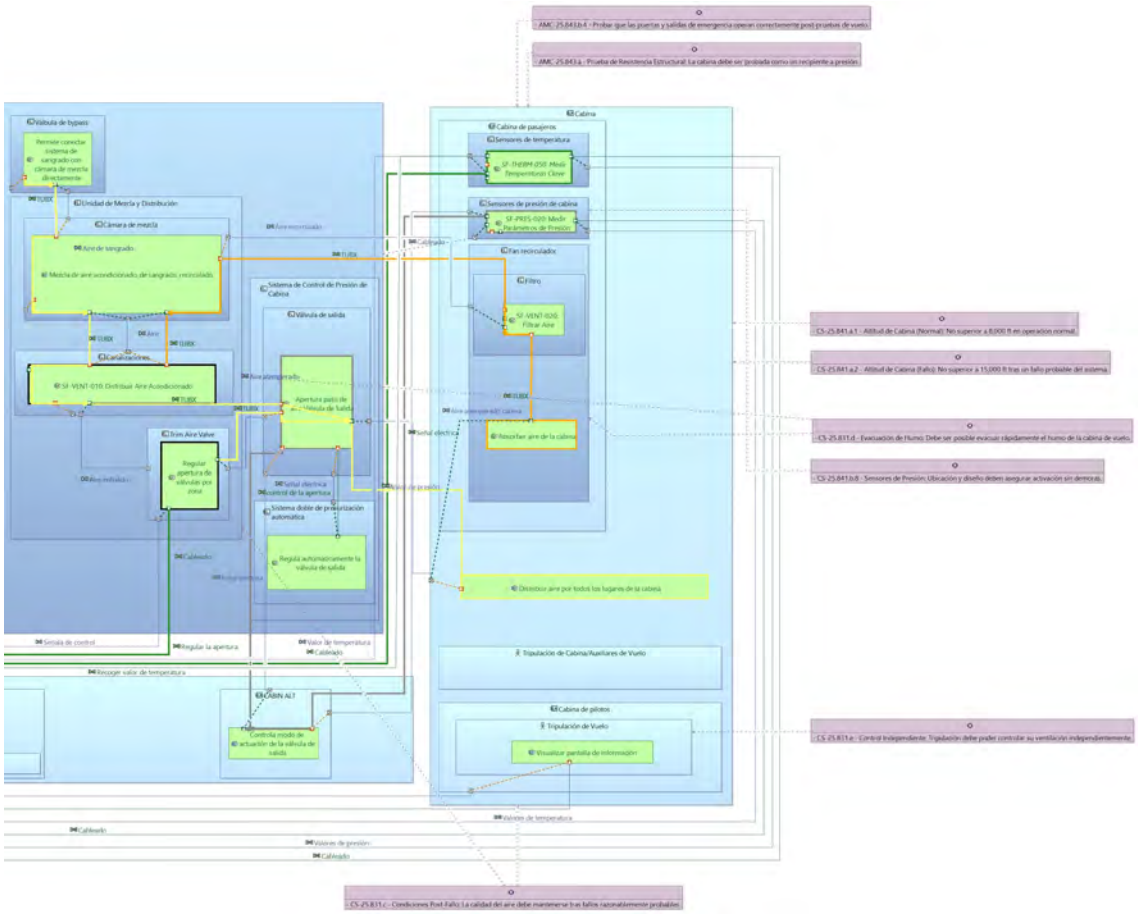

Figura 3.43 Esquema LAB: Cabina.

3.5.4 Análisis Físico

Una vez modelada la Arquitectura Lógica, el Análisis Físico se encarga de transformar esa solución conceptual en una arquitectura concreta y viable, detallando cómo se construirá e implementará el sistema. Para ello, se seleccionan tecnologías y se definen los componentes físicos. Dado que no se ha realizado un estudio de mercado para la selección de componentes, se considerará cada componente como físico.

A nivel funcional, no se realizarán cambios, ya que todo está definido en la capa lógica. El principal cambio en esta capa física será la introducción de redundancias, además de la elección de componentes. En nuestro caso, se incluirán dos sistemas PACK, dos válvulas OFV y dos sensores de presurización de cabina. Esta decisión se justifica por un análisis previo de árbol de fallos (FTA) que se expondrá más adelante que busca cumplir con el requisito de probabilidad de fallo catastrófico, el cual exige redundancias en los sistemas de presurización, cuyo fallo podría poner en riesgo la vida de los ocupantes. La inclusión de un tercer PACK, aunque teóricamente reduciría aún más la probabilidad de fallo catastrófico, se considera una solución ineficiente desde la perspectiva del coste y el peso, tal y como se analizará cuantitativamente en el estudio de compromiso (trade-off). Todo esto está explicado en la **sección 3.6.2.**

En la industria real el ingeniero de sistema tendría acceso a un catálogo de piezas con sus proveedores proporcionado por la compañía encargada del proyecto. Aquí tendría que hacer un estudio comparativo, por ejemplo, valorando el rendimiento en simulaciones de las distintas piezas y eligiendo *ad hoc*. No se puede hilar tan fino por lo que simplemente se asume el significado

real y técnico que tienen los componentes físicos de nodo y los enlaces físicos, sin llegar de hacer referencia a uno real y determinado.

En **Figura 3.44**, **Figura 3.45** y **Figura 3.46** se muestran las distintas partes del diagrama PAB sin las funciones mapeadas:

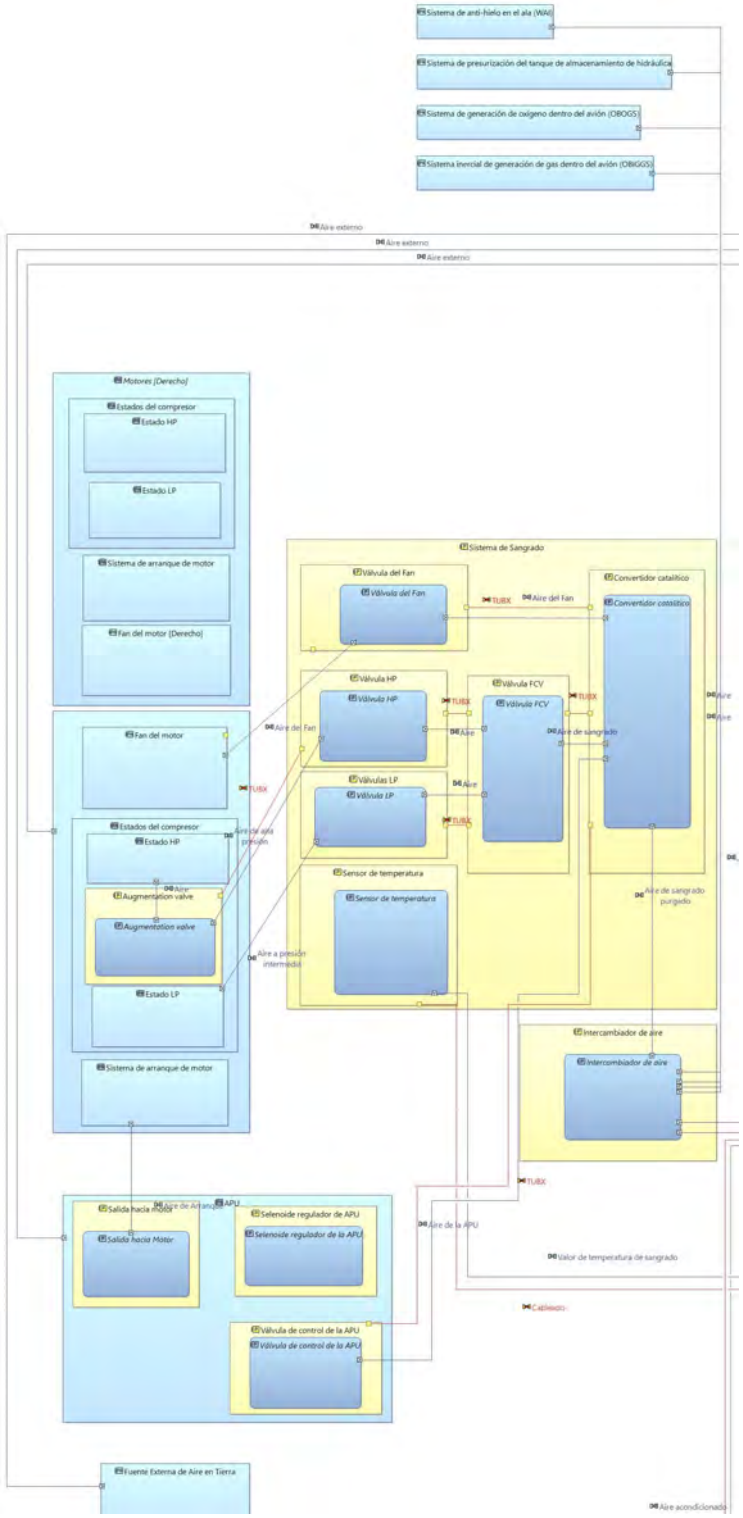

Figura 3.44 Primera representación gráfica del PAB.

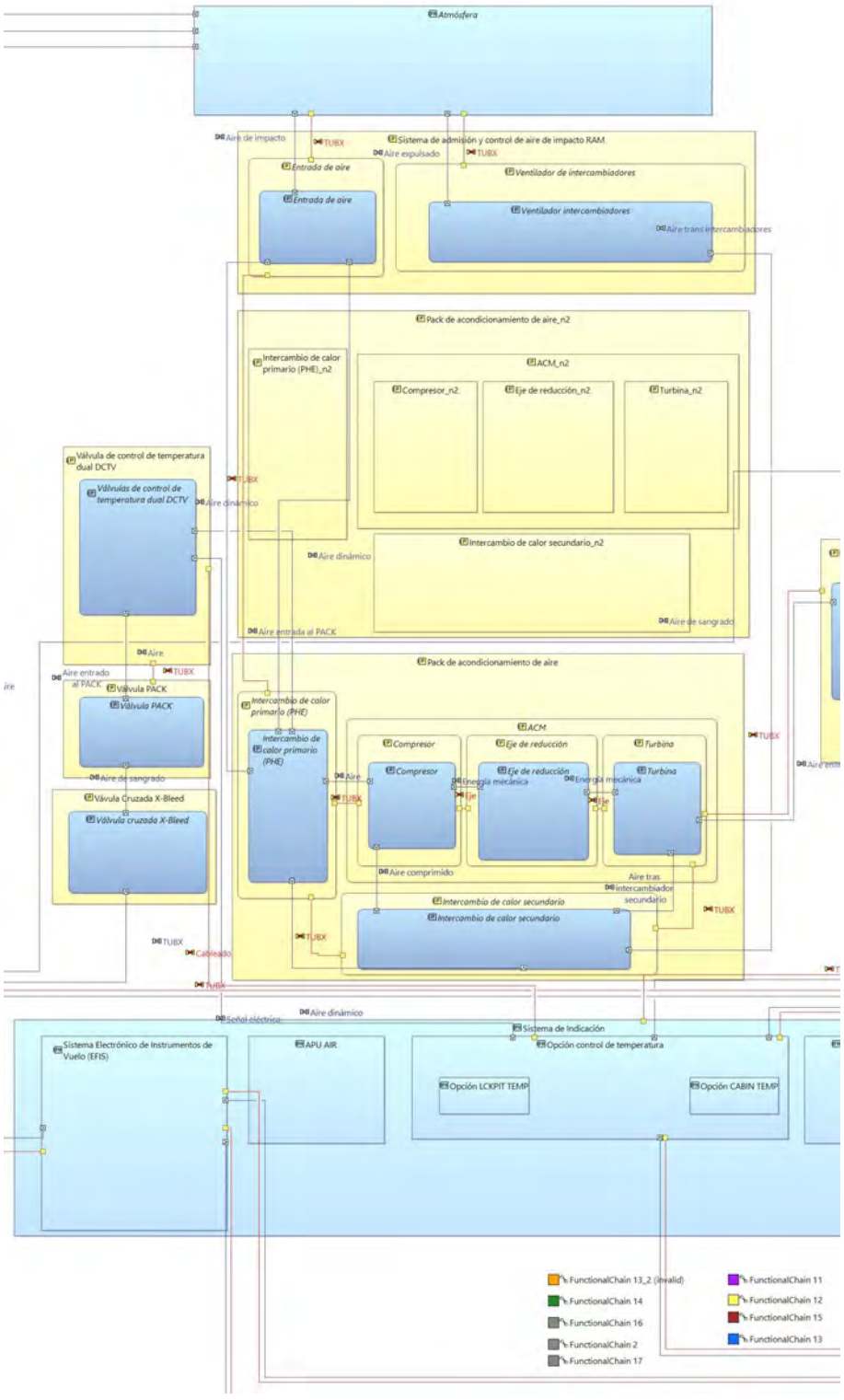

Figura 3.45 Segunda representación gráfica del PAB.

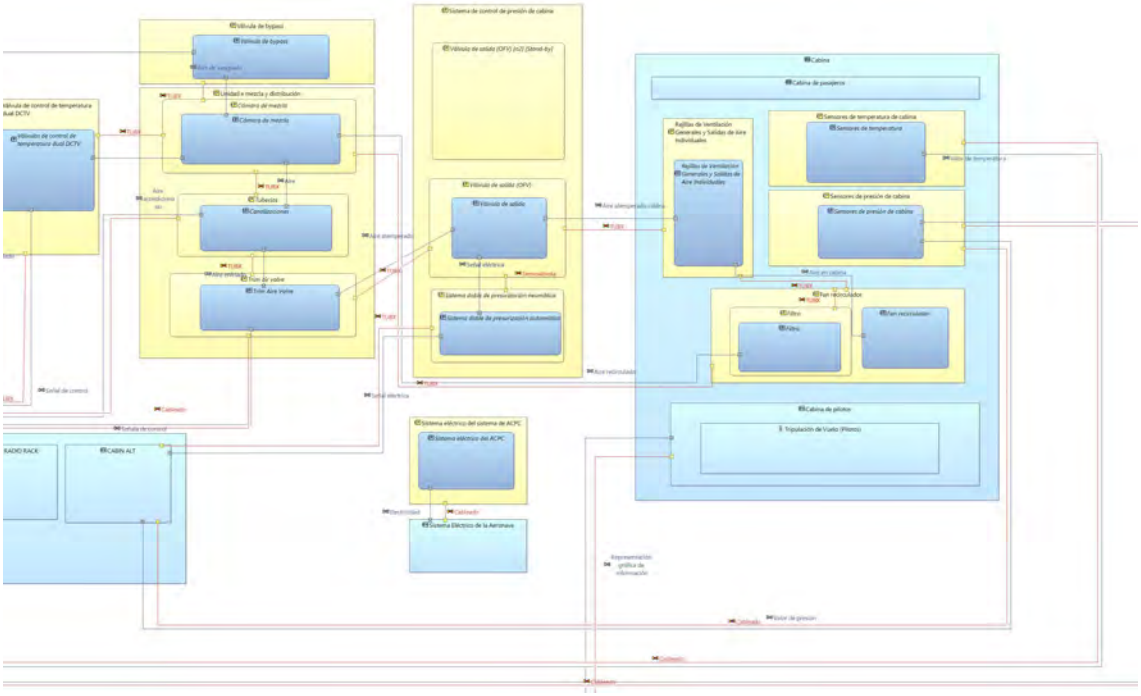

Figura 3.46 Tercera representación gráfica del PAB.

En **Figura 3.47**, **Figura 3.48** y **Figura 3.49** se muestran las distintas partes del diagrama PAB con las funciones mapeadas y sus correspondientes cadenas funcionales:

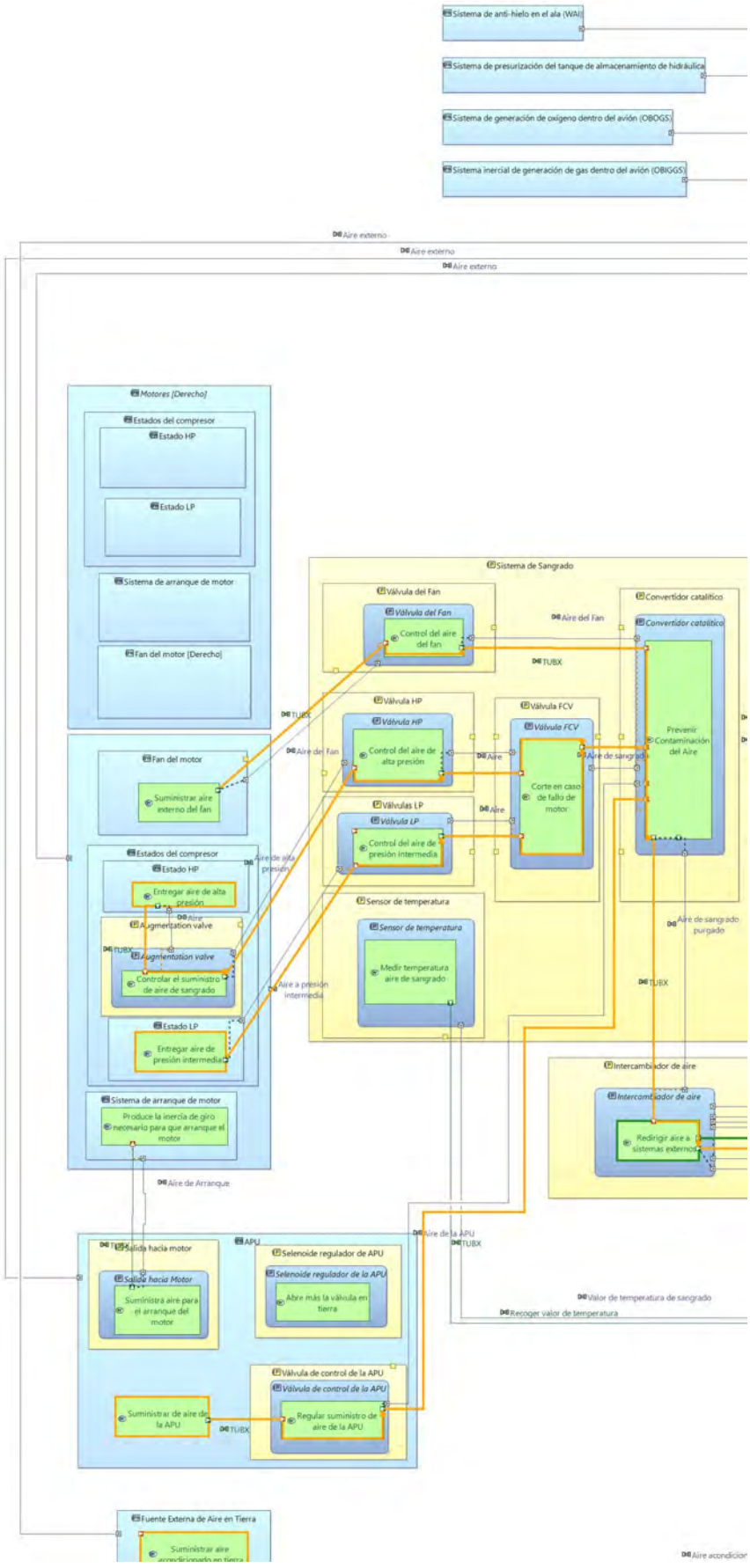

Figura 3.47 Primera representación gráfica del PAB con análisis de funciones.

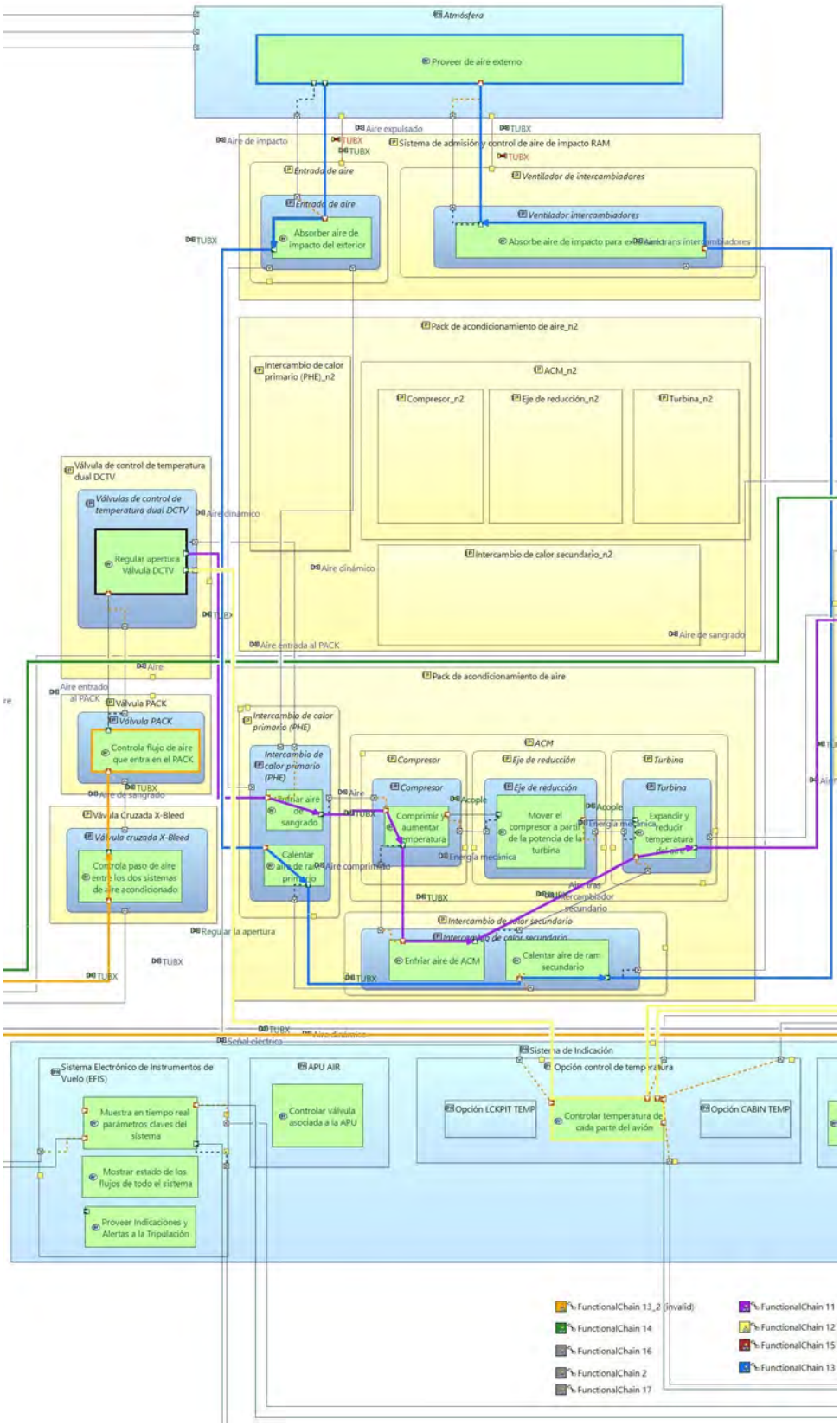

Figura 3.48 Segunda representación gráfica del PAB con análisis de funciones.

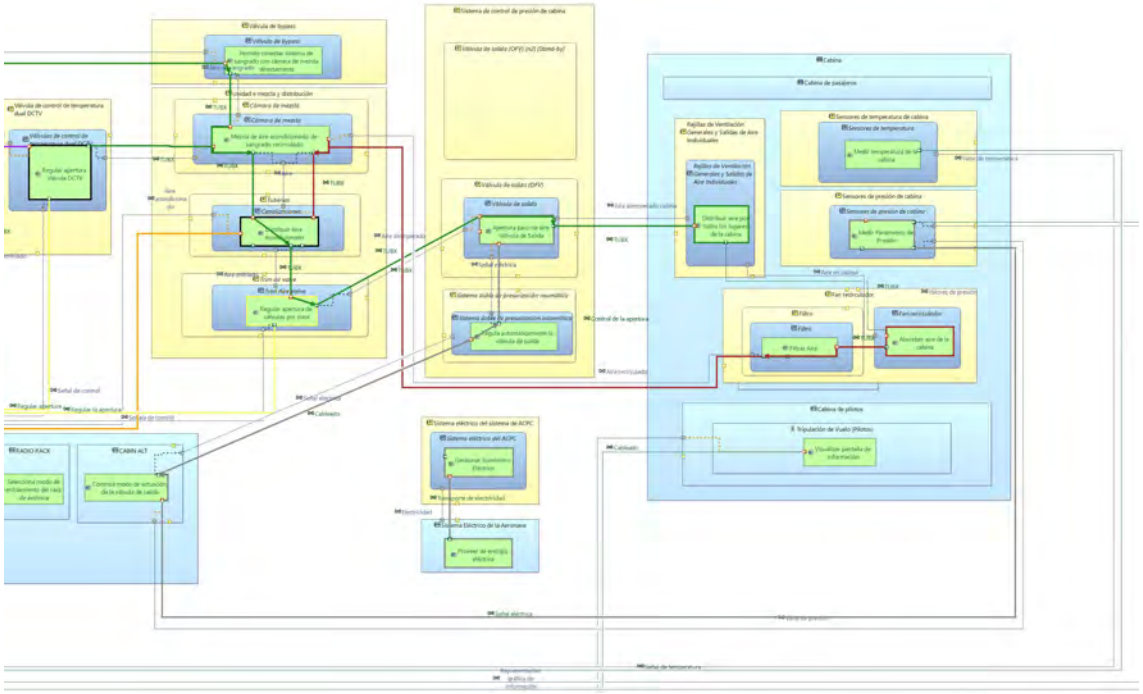

Figura 3.49 Tercera representación gráfica del PAB con análisis de funciones.

3.6 Integración con Simulación para Validación de Requisitos

El principal objetivo que se va a proponer es la comprobación de dos requisitos del diseño que tendrán asociado un valor. Se deberá comprobar que ese valor queda dentro del intervalo definido por el requisito a través de diferentes escenarios simulados. En base al resultado se podrá comprobar si con el dimensionamiento planteado y los valores definidos se cumpliría el requisito o no.

3.6.1 Modelo 1: Integración con Simulink para Verificación de Requisitos

En nuestro caso, se va a intentar verificar el requisito **CS-25.841.a.1**. Para ello, se configurará la integración entre el modelo de arquitectura MBSE en Capella y un modelo de simulación en Simulink, demostrando así el potencial de MBSE para validar y verificar requisitos conjuntamente con otros programas mediante unas pocas líneas de Python.

Requisito bajo estudio: CS 25.841.a.1

> **CS 25.841 Requisitos para Cabinas Presurizadas**
> Esta sección establece los criterios esenciales que deben cumplir las cabinas y compartimentos presurizados destinados a la ocupación.
>
> **Apartado (a) - Niveles de Altitud de Cabina:** *Condiciones Normales de Operación: Se exige que la cabina mantenga una altitud de presión no superior a 2.438 m (8.000 ft) cuando la aeronave opere a su altitud máxima certificada.*

Metodología de Integración

La integración se ejecutará con **Python4Capella**, un *add-on* gratuito de Capella con funcionalidades que abarcan desde la importación de BOM, PBS, FBS con sus estructuras jerarquizadas, hasta la exportación de información sobre los diagramas y componentes lógicos.

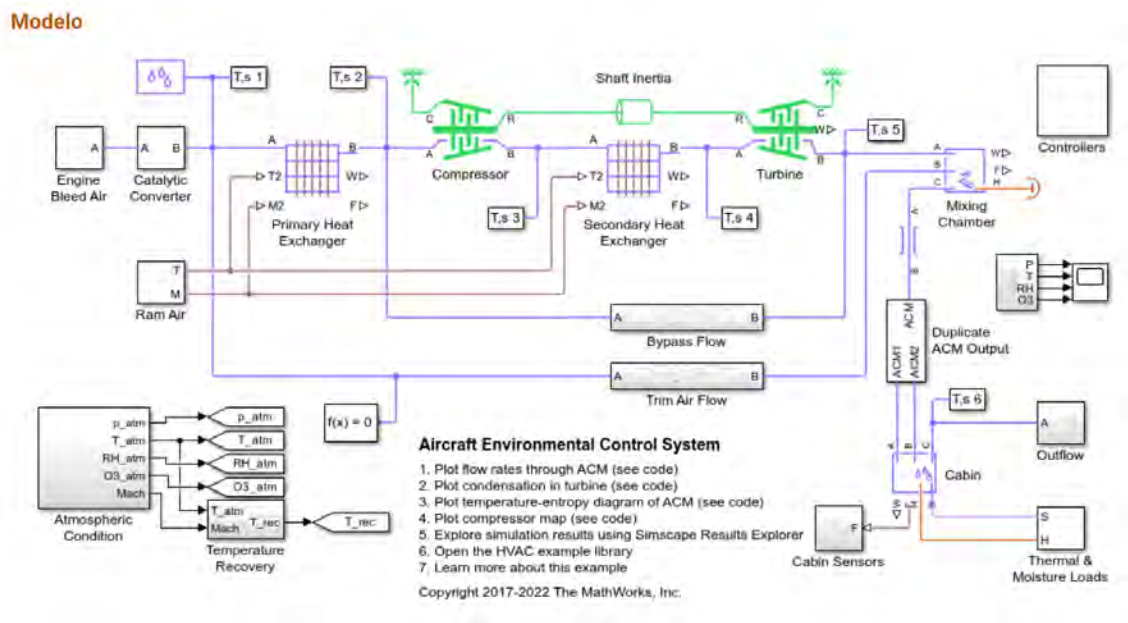

Figura 3.50 Modelo ECS realizado en Simulink.

El modelo de referencia es una simulación completa del Sistema de Control Ambiental (ECS) disponible en la página de MATLAB (**Figura 3.50**), el cual regula la presión, temperatura, humedad y ozono (O3). Este modelo tiene asociado un perfil de vuelo (despegue, crucero y aterrizaje) y se utilizará como una *caja negra*: se cogerán las dimensiones y magnitudes que sean de interés, se parametrizarán y se enviarán los valores que se quieran para el diseño al modelo de Simulink. Se desarrollará toda la simulación consistente en un vuelo completo con distintas fases de subida y bajada. Se nos devolverán ciertos valores de vuelta, outputs.

Para definir las propiedades y parámetros asociados a los componentes del modelo se hará uso de otro *add-on* llamado **PMVT**. Los parámetros se definirán en la capa lógica del modelo (**Figura 3.51** y **Figura 3.52**), ya que el modelo de simulación utilizado está simplificado a un solo PACK y lo que interesa es representar el funcionamiento mismo. Cabe señalar que el valor del requisito `p_8000m` (valor límite para que se cumpla la normativa) se ha asociado directamente a la cabina en Capella y no al requisito mismo. Lo ideal es que este valor esté en el propio requisito, pero debido a que estamos en una fase de aprendizaje mientras desarrollamos el proyecto, solo se ha conseguido llamar desde Python4Capella a los valores de PVMT que estuviesen en componentes o actores.

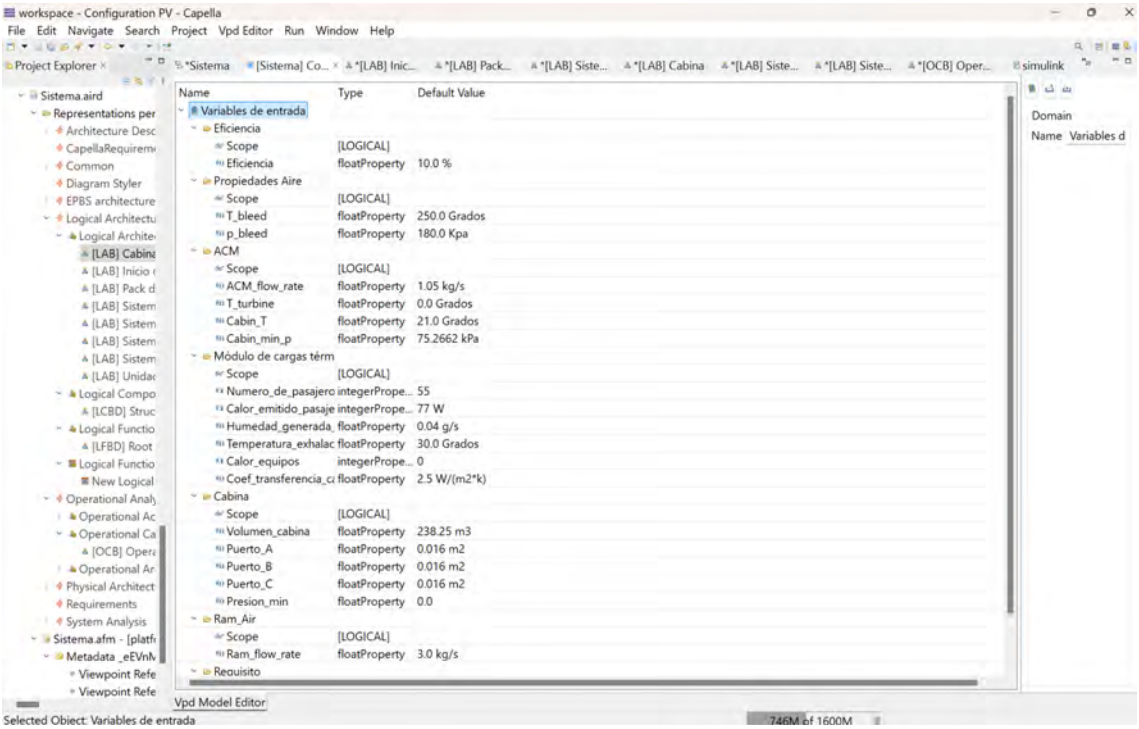

Figura 3.51 Pestaña general de propiedades de PVMT.

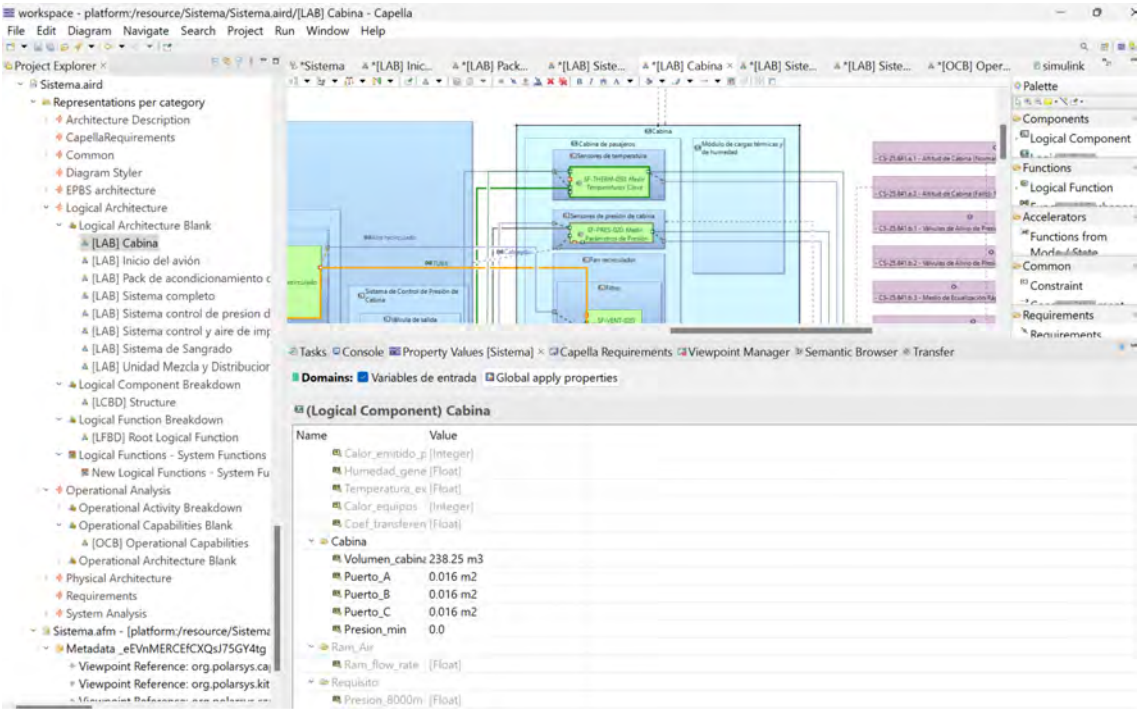

Figura 3.52 Pestaña de un componente específico con sus propiedades en PVMT y los valores establecidos por el ingeniero de arquitectura.

Proceso del Script de Integración (Python4Capella y MATLAB)

Se conecta Capella con el modelo de Simulink mediante un script (referenciado en el Apéndice B) cuyo proceso se desglosa en los siguientes pasos clave:

1. **Inicialización de Entornos:**
   - Se establece la conexión con la API de Capella (**Python4Capella**).
   - Se inicia una instancia del motor de MATLAB (`matlab.engine`).
2. **Extracción de Parámetros del Modelo (Capella):**
   - Se abre el modelo de arquitectura (`.aird`).
   - Se itera a través de los Componentes Lógicos y Actores Lógicos y se utiliza la API de **PVMT** para leer los valores de las propiedades de diseño (ej. `T_bleed`, `p_bleed`, cargas térmicas, etc.).
   - Se extrae el valor del requisito `Presion_8000m`, que representa la presión mínima requerida.
3. **Ejecución de la Simulación (MATLAB):**
   - Se transfieren todos los parámetros de Capella al espacio de trabajo de MATLAB.
   - Se ejecuta un script de MATLAB (`.m`) que corre la simulación en Simulink.
   - Se recupera el resultado clave: la presión mínima alcanzada en la cabina (`presion_cabina_min`).
4. **Verificación del Requisito y Actualización del Modelo (Capella):**
   - Se compara el resultado de la simulación (`presion_actual`) con el valor requerido (`presion_requerida`).
   - Se busca el requisito por su nombre (`CS-25.841.a.1`) y se actualiza su prefijo a "REQUISITO NO SATISFECHO" o "REQUISITO SATISFECHO". Como se ve en la **Figura 3.53**.
   - Se guardan los cambios en el modelo de Capella.

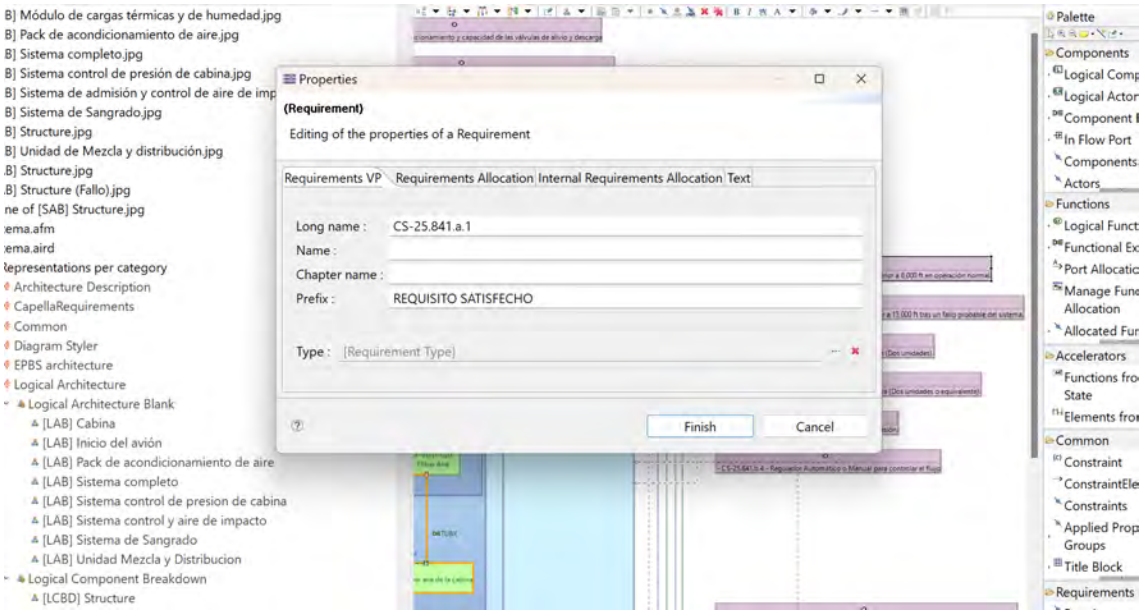

Figura 3.53 El requisito se muestra como satisfecho en el atributo prefijo.

5. **Finalización:** Se cierra la conexión con el motor de MATLAB.

3.6.2 Modelo 2: Análisis de Fiabilidad (FTA) y Coste en MATLAB

En este segundo modelo (referenciado en el Apéndice D), nuestro objetivo es validar el siguiente requisito extraído del **CS 25.1309: Equipos, Sistemas e Instalaciones**:

> **Gestión de Fallos:** Los sistemas y componentes asociados, tanto individualmente como en relación con otros sistemas, deben diseñarse de forma que cualquier condición de fallo catastrófico sea **extremadamente improbable** y no sea consecuencia de un **único fallo**.

De esta manera, se ha considerado como fallo catastrófico la **pérdida de la presurización de la cabina**, lo que podría acabar con la vida de todos los pasajeros y la tripulación de vuelo.

Para el desarrollo de este modelo, la contribución de Gemini ha sido fundamental, permitiendo identificar rápidamente cómo el modelado de la arquitectura podría estar limitado por las directrices de los ingenieros RAMS[6]. El script (referenciado en el Apéndice C) se estructura en las siguientes secciones.

Sección 0: Configuración del Modelo

Se establece la parametricidad del modelo. Las variables `N_PACKs`, `N_OFV` y `N_CPC` actúan como interruptores que definen la arquitectura del sistema a analizar.

Sección 1: Parámetros de Entrada

Define las constantes del análisis: tasas de fallo (probabilidad de fallo por hora) y costes (adquisición y consecuencia del fallo). Como se ve en la **Figura 3.54**. El ingeniero RAMS debería acceder a estándares como **MIL-HDBK-217** o **FIDES** para estimar estas probabilidades.

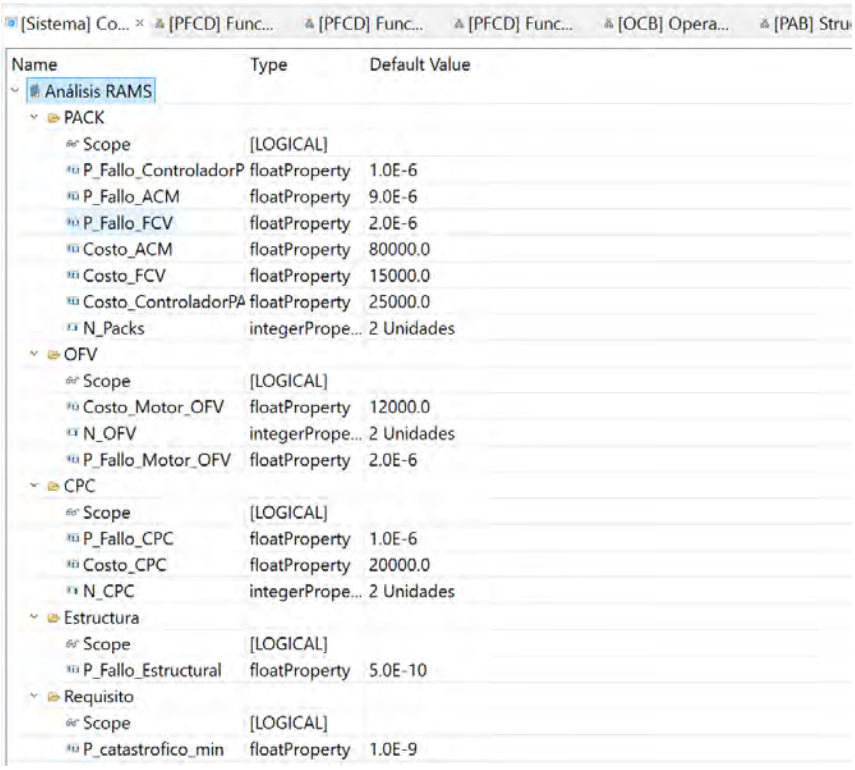

Figura 3.54 Propiedades parametrizadas para nuestro análisis de FTA.

[6] En el proyecto se ha considerado valores promedios para el tipo de componentes que se trataban, no se ha realizado un estudio exhaustivo

Sección 2: Cálculo Analítico de Probabilidad (Bottom-Up)

Es la implementación matemática del Árbol de Fallos (FTA). Se calcula la probabilidad de fallo de los componentes individuales y se combinan (con puertas lógicas OR y AND) para obtener la probabilidad del *Top Event*: la pérdida de presurización.

El Árbol de Fallos para la Pérdida de Presurización se inicia con este evento superior, que representa el estado final indeseable de la aeronave. Inmediatamente debajo, una puerta OR establece que este fallo ocurre si se da cualquiera de sus dos causas principales: el fallo funcional del sistema (Salida > Entrada) o el fallo físico del contenedor (Fallo Estructural). Esta primera bifurcación separa lógicamente los problemas de control y equilibrio de aire de una brecha física en el fuselaje, tratando esta última como una causa raíz.

Profundizando en la rama funcional, el evento Salida > Entrada se desglosa a su vez mediante otra puerta OR, indicando que el desequilibrio se produce por un fallo en el suministro de aire (Pérdida Total PACKs) o por un fallo en la regulación de su evacuación (Fallo Control Salida). De esta manera, el análisis distingue claramente entre no poder introducir suficiente aire y no poder controlar el aire que se escapa, aislando las dos funciones críticas del sistema de presurización.

Finalmente, el árbol conecta con los fallos de hardware a través de la lógica de redundancia. El evento Pérdida Total PACKs es la salida de una puerta AND, lo que significa que TODOS los PACKs deben fallar para que el suministro cese. A su vez, el fallo de cada PACK individual es el resultado de una puerta OR interna, ya que basta con que falle uno de sus subcomponentes clave (ACM, válvula, etc.). Es en esta última capa donde se encuentran los eventos básicos, conectando el fallo general de la aeronave con las averías específicas y cuantificables de los componentes raíz.

Sección 3: Visualización del Árbol de Fallos

Construye y dibuja dinámicamente un diagrama del árbol de fallos utilizando la funcionalidad `digraph` de MATLAB. Como se ve en la **Figura 3.55**.

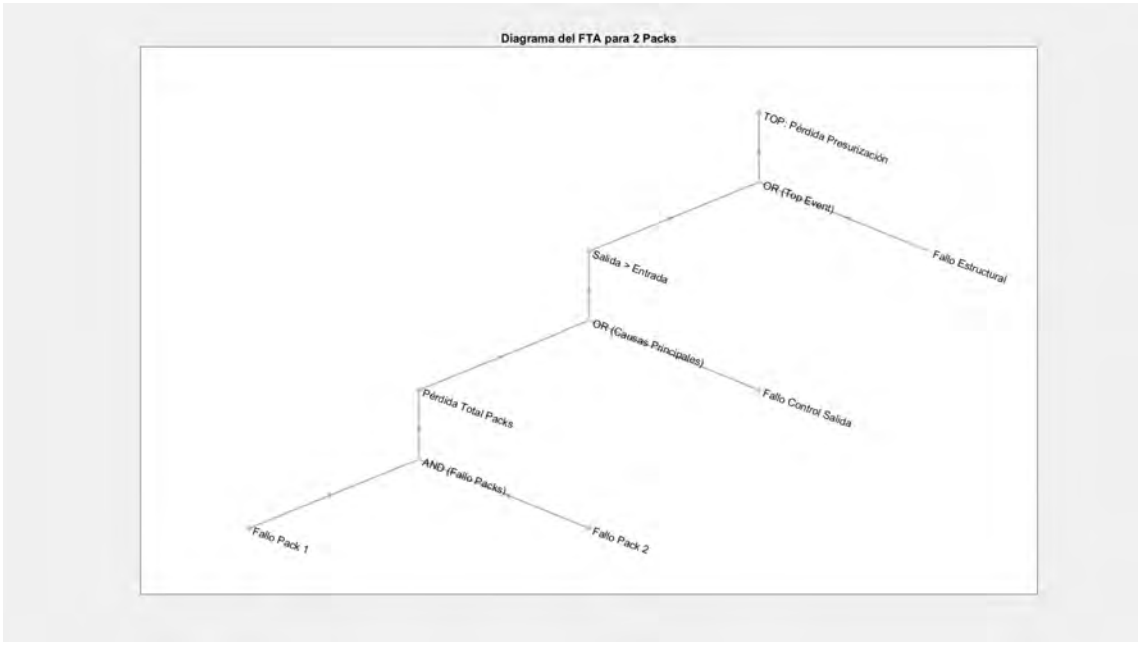

Figura 3.55 Representación gráfica del FTA.

Sección 4: Simulación de Monte Carlo

Esta sección (comentada en el script original por su coste computacional) verifica el resultado analítico mediante un método estocástico. Simula millones de "horas de vuelo" y utiliza números

aleatorios para determinar si los componentes fallan. La probabilidad del *Top Event* se estima como el cociente entre simulaciones con fallo y el total de simulaciones.

Sección 5: Análisis de Coste y Compromiso Multidimensional

Es la culminación del script. Combina los análisis de fiabilidad y coste para crear una herramienta de decisión. El código prueba cada combinación posible de arquitectura (ej., 1 PACK, 1 OFV, 2 CPCs), recalcula la probabilidad de fallo y el coste, y genera un gráfico de Coste vs. Fiabilidad. Este gráfico (**Figura 3.56**) permite visualizar directamente qué configuraciones ofrecen la mayor fiabilidad para un presupuesto determinado.

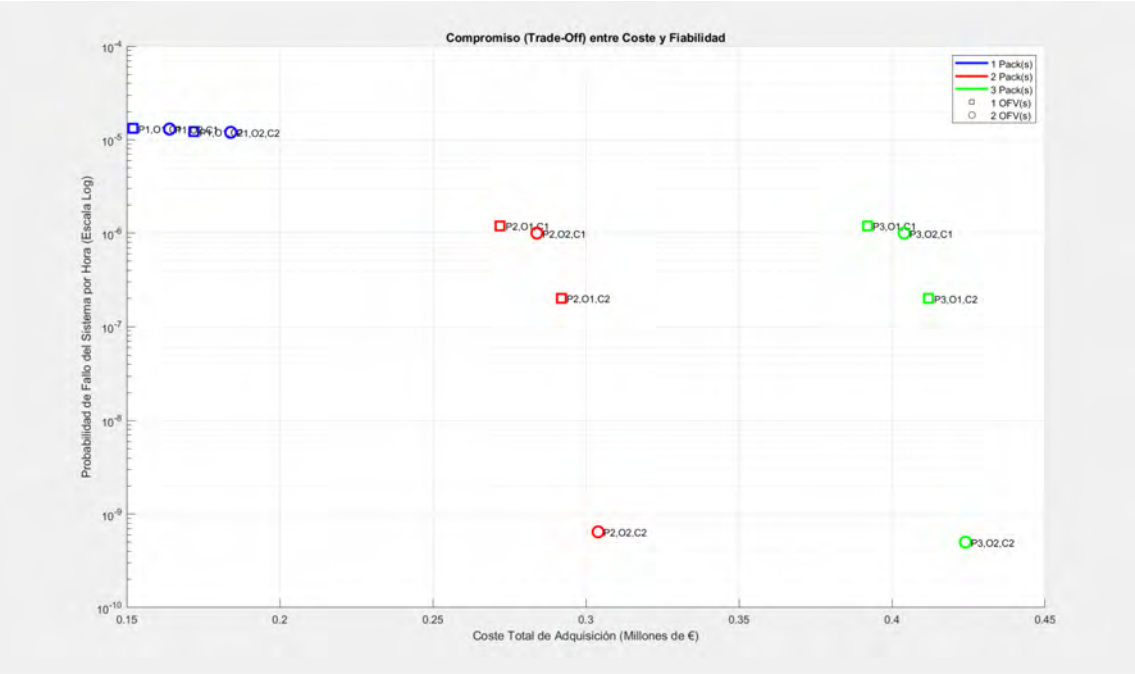

Figura 3.56 Análisis de trade-off: Nº de componentes/ Costo total / Probabilidad de fallo del evento total.

Hipótesis y Simplificaciones Realizadas

- **Independencia de Fallos:** Se asume que el fallo de un componente no influye en la probabilidad de que otro falle.
- **Tasas de Fallo Constantes:** Se utilizan probabilidades fijas por hora (modelo exponencial), ignorando efectos de desgaste.
- **Aditividad de Probabilidades (Puerta OR):** El cálculo $P(A \cup B) \approx P(A) + P(B)$ es una aproximación válida porque las probabilidades de fallo individuales son muy pequeñas. La fórmula exacta, $P(A \cup B) = P(A) + P(B) - P(A \cap B)$, se simplifica despreciando el término de la intersección.
- **Componentes Idénticos:** Se asume que todos los componentes de un mismo tipo (ej. todos los PACKs) son idénticos.
- **Fallo de ADIRU Despreciable:** La probabilidad de fallo del ADIRU se establece explícitamente en cero para simplificar el análisis.
- **Modelo de Sistema Simplificado:** El árbol de fallos es una representación de alto nivel y no modela todas las interacciones complejas.

Integración en Capella

Los pasos que se han seguido para esta segunda integración planteada en el **Script C** son parecidos a los del modelo anterior, pero con nuevas variables de entrada.

1. Conexión inicial

Básicamente, lo primero que hace el script es arrancar y conectarse a dos cosas a la vez:

- Al modelo de **Capella**, para poder leerlo y escribir en él.
- Al motor de **MATLAB**, para poder pedirle que haga cálculos.

2. Extracción de datos de Capella

Una vez conectado, el script se pone a buscar en tu modelo de Capella (**Figura 3.57**). Recorre todos los `LogicalComponent` y `LogicalActor` que están modelados. De cada uno de ellos, se extraen sus propiedades (los *Property Values*), que son los datos clave del diseño:

- **Probabilidades de fallo**: como `P_Fallo_ACM`, `P_Fallo_ControladorPACK`, etc.
- **Costes**: como `Costo_ACM`.
- **Cantidades**: como el número de PACKs (‘N_PACKs‘).

La idea es que el modelo de Capella sea la **fuente de verdad única**: si se cambia un valor en el modelo, el script lo cogerá de ahí directamente sin que se tenga que tocar el código.

Figura 3.57 Se extraen los parámetros definidos en Capella.

3. Simulación en MATLAB

Con todos los datos ya extraídos, el script llama a la función `run_matlab_simulation`. Esta función hace dos cosas:

- Le pasa a MATLAB todos los parámetros que ha sacado de Capella (probabilidades, costes, etc.).

- Le ordena a MATLAB que ejecute el script `FTA_Aire_acondicionado.m`. Este **Script D** es el que hace el trabajo duro: un **Análisis de Árbol de Fallos (FTA)** para calcular la probabilidad de fallo total del sistema (**Figura 3.58**).

```
Extracci├│n de par├ímetros finalizada.

--- Iniciando simulaci├│n en MATLAB ---
--- CONFIGURACIÓN DEL SISTEMA ---
Modelo configurado para:
- 2 Packs de Aire Acondicionado
- 2 V lvulas de Salida (OFV)
- 2 Controladores de Presi¢n (CPC)

--- AN╡LISIS CUANTITATIVO ---
Probabilidad de fallo de TODOS los Packs (AND): 1.44e-10
PROBABILIDAD DEL TOP EVENT (Calculada) POR HORA: 6.4801e-10

--- VISUALIZACIÓN ---
Construyendo grafo del FTA din micamente...
Visualizaci¢n generada.

--- AN╡LISIS DE COSTE ---
Coste de Adquisici¢n de 1 Pack: 120000.00 □
Coste Total de Adquisici¢n del Sistema: 304000.00 □
Coste Esperado de Fallo por Hora (Mantenimiento por Fiabilidad): 7.05 □/hora
```

Figura 3.58 Se simula el modelo en capella y se obtienen los resultados.

4. Verificación y actualización del modelo

Cuando MATLAB termina, le devuelve al script de Python el resultado final: la probabilidad de fallo del sistema (‘p_actual‘). Entonces, el script:

- Busca en Capella el requisito llamado `CS-25.1309.c.1`.
- Como se ve en la **Figura 3.59**,se compara el resultado de la simulación con el valor límite que tiene ese requisito (‘P_catastrofico_min‘). La condición matemática que comprueba es:

$$P_{\text{actual}} \leqslant P_{\text{catastrófico\_min}}$$

- **Actualiza el modelo**: Si la condición se cumple, va al requisito en Capella y le añade el prefijo **"REQUISITO SATISFECHO"**. Si no, le pone **"REQUISITO NO SATISFECHO"**. Como ya se mostró en el anterior caso (**Figura 3.53**). Así, el resultado del análisis queda registrado directamente en el modelo de diseño.

```
--- AN╡LISIS DE COMPROMISO MULTIDIMENSIONAL ---
Calculando el espacio de dise¤o para Packs, OFVs y CPCs...
Espacio de dise¤o calculado:
    Configuracion    N_Packs    N_OFV    N_CPC     Costo      Probabilidad
    _____________    _______    _____    _____    ________    ____________

    "P1,O1,C1"         1          1        1      1.52e+05      1.41e-05
    "P1,O1,C2"         1          1        2      1.72e+05     1.4001e-05
    "P1,O2,C1"         1          2        1      1.64e+05     1.2101e-05
    "P1,O2,C2"         1          2        2      1.84e+05     1.2001e-05
    "P2,O1,C1"         2          1        1      2.72e+05     2.1006e-06
    "P2,O1,C2"         2          1        2      2.92e+05     2.0006e-06
    "P2,O2,C1"         2          2        1      2.84e+05     1.0065e-07
    "P2,O2,C2"         2          2        2      3.04e+05     6.4801e-10
    "P3,O1,C1"         3          1        1      3.92e+05     2.1005e-06
    "P3,O1,C2"         3          1        2      4.12e+05     2.0005e-06
    "P3,O2,C1"         3          2        1      4.04e+05      1.005e-07
    "P3,O2,C2"         3          2        2      4.24e+05     5.0401e-10

Simulaci├│n realizada con ├Éxito.
A continuaci├│n se mostrar├ín los par├ímetros output calculados:
P_Top_Event_Calculada: 6.4801e-10

--- Actualizando el modelo de Capella con los resultados ---
Evaluando Requisito ID: cbfced5c-c7bd-4fd7-bdc1-34b92cb84f77 - Nombre: EASA-CS25
Evaluando Requisito ID: _58664d0b-f96f-4837-88dd-f5b023c3c94d - Nombre: EASA-CS25.CS
Evaluando Requisito ID: _4a9f8653-e740-48ea-a035-21ac8174ac8a - Nombre: CS-BOOK1.D
Evaluando Requisito ID: _3a6fec66-9471-4f03-92b7-905f728d3c38 - Nombre: CS-25.1309
Evaluando Requisito ID: _2c3dcd02-b087-4cbe-bdda-d91619e19152 - Nombre: CS-25.1309.c
Evaluando Requisito ID: _047755f3-7d9f-42f1-8864-a7c6cdf98337 - Nombre: CS-25.1309.c.1
  -> Requisito de probabilidad encontrado. Verificando condici├│n...
  -> RESULTADO: SATISFECHO (Probabilidad simulada: 6.48E-10, Requerida <= 1.00E-09)
Evaluando Requisito ID: cf1af2e2-c674-414e-99e9-c4838aa8f28b - Nombre: CS-25.1309.c.2
Evaluando Requisito ID: _0272e7bc-7e58-4c0b-9cbf-4186c5979515 - Nombre: CS-25.1309.c.3
Evaluando Requisito ID: _5ec51010-ea05-48e3-a0d8-03f99a81bb91 - Nombre: CS-25.1309.a
```

Figura 3.59 Se compara el resultado obtenido de simulación para el requisito a analizar.

4 Discusión

4.1 Análisis de la Experiencia de Aplicación de MBSE

La aplicación práctica de la Ingeniería de Sistemas Basada en Modelos (MBSE) revela una tecnología con un notable potencial y un largo recorrido por delante. A pesar de que en la actualidad parte de los ingenieros de sistemas a menudo la perciba como una iniciativa de I+D, otra gran parte considera que la arquitectura de un sistema constituye el pilar central de cualquier diseño, ya que en torno a ella gira todo el ciclo de vida del desarrollo de un producto, erigiéndose como la columna vertebral de esta disciplina. Esto subraya el imperativo estratégico de que el MBSE sea elevado a una disciplina de primer orden dentro de cualquier marco de PLM (Product Lifecycle Management).

En el contexto actual, donde la competitividad del mercado exige una adaptación constante a las necesidades de un consumidor cada vez más exigente, la adopción de MBSE no es una opción, sino una necesidad. Solo mediante la gestión rigurosa de requisitos, su validación y verificación continuas, y la trazabilidad completa que ofrece un modelo, es posible gestionar la complejidad inherente a los sistemas de ingeniería modernos.

Aunque el MBSE es una disciplina relativamente reciente y su enseñanza aún no está generalizada en el ámbito universitario, es previsible que su adopción crezca exponencialmente. Su potencial no solo reside en su sistematicidad y en la capacidad que tiene para integrar distintas empresas que participan en un mismo proceso de ingeniería bajo un prisma común, sino también en su valor pedagógico: el desarrollo de capacidades y su vinculación con modelos funcionales permite una comprensión rápida e intuitiva de sistemas complejos, al simplificar la realidad para centrarse en las variables de interés.

El proceso de aprendizaje de MBSE puede ser ágil cuando se realiza en el contexto de un equipo de trabajo consolidado y en proyectos reales. Sin embargo, el autoaprendizaje puede resultar un desafío, dependiendo en gran medida del acceso a materiales de formación especializados, que a menudo son de pago. En este sentido, el hecho de que herramientas como Capella sean de código abierto ha acelerado significativamente su visibilidad y adopción, permitiendo a las empresas experimentar con la metodología en proyectos piloto que han demostrado ser exitosos. Adicionalmente, la flexibilidad para desarrollar extensiones (Add-ons) en Capella facilita la adaptación de la arquitectura a una vasta gama de problemas de ingeniería.

4.2 Valor Aportado por MBSE frente a Enfoques Tradicionales

4.2.1 Dificultades en el proceso de adopción del MBSE en un proyecto

La transición hacia la Ingeniería de Sistemas Basada en Modelos (MBSE) representa un cambio paradigmático significativo respecto a los enfoques tradicionales basados en documentos (DBSE).

Si bien los beneficios potenciales de MBSE –como la mejora en la gestión de la complejidad, la detección temprana de errores, la coherencia del diseño y la facilitación de la comunicación interdisciplinar– son ampliamente reconocidos, su adopción efectiva se enfrenta a un obstáculo considerable: la necesidad de una inversión inicial sustancial cuyos retornos tangibles a menudo no se materializan hasta fases avanzadas del proyecto o incluso en proyectos futuros.

En la **Figura 4.1** se muestra una comparación conceptual de la inversión en MBSE (curva azul, alto coste inicial decreciente, menor costo final) frente a los costes de ingeniería basada en documentos (curva roja, bajo coste inicial creciente, mayor coste final) a lo largo del ciclo de vida del sistema [Hysko, 2023].

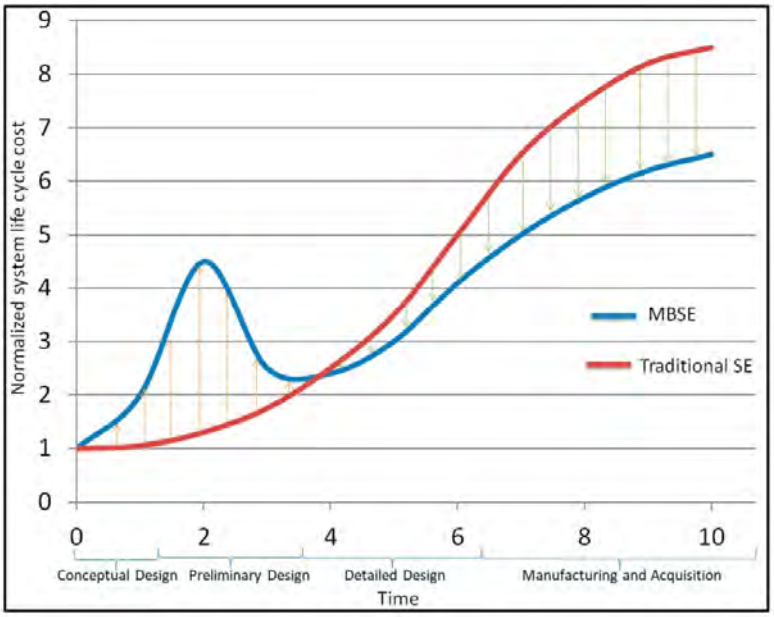

Figura 4.1 Comparación conceptual de la inversión en MBSE frente al enfoque tradicional.

4.2.2 Naturaleza de la Inversión Inicial:

La implementación de MBSE requiere un desembolso inicial significativo que abarca múltiples áreas:

- **Costes de Herramientas:** Adquisición de licencias de software de modelado (ej. Capella, Cameo, Rhapsody, Enterprise Architect), herramientas de gestión de requisitos compatibles (ej. DOORS), plataformas PLM (ej. Teamcenter) y posibles herramientas de simulación o análisis que se integren con los modelos.
- **Formación y Desarrollo de Competencias:** Es fundamental invertir en la capacitación de los ingenieros, no solo en el manejo técnico de las herramientas, sino también en la comprensión profunda de la metodología MBSE elegida (ej. Arcadia, SysML estándar) y en cómo aplicarla eficazmente en el contexto específico del proyecto y la organización. Esto implica tiempo y recursos dedicados a la formación y a superar la curva de aprendizaje inicial.
- **Esfuerzo de Modelado Inicial:** La creación de los modelos fundacionales, la definición de la arquitectura del sistema, la captura inicial de requisitos y funciones, y el establecimiento de las librerías de componentes reutilizables requieren un esfuerzo considerable por parte de un equipo dedicado de ingenieros de sistemas y arquitectos. Este esfuerzo inicial es intrínsecamente más intensivo que la simple redacción de documentos iniciales en un enfoque DBSE.

- **Adaptación de Procesos y Cultura:** MBSE no es solo una herramienta, sino un cambio en la forma de trabajar. Requiere adaptar los procesos de ingeniería existentes, definir nuevos roles (como el "Model Owner" o "Model Curator") y fomentar una cultura organizacional que valore y confíe en el modelo como fuente central de verdad.

4.2.3 El Retorno Diferido y el Riesgo de Abandono:

La principal dificultad radica en que los beneficios más significativos de MBSE, especialmente los relacionados con la eficiencia y la reducción de costes, no son inmediatos. Si bien la detección temprana de errores puede generar ahorros durante el mismo proyecto, el verdadero potencial de MBSE se desbloquea a medio y largo plazo. Esto se debe a varios factores:

- **Madurez del Modelo:** Los modelos necesitan tiempo para madurar, ser validados y alcanzar un nivel de detalle y coherencia suficiente para soportar análisis complejos y la generación automática de artefactos.
- **Curva de Aprendizaje Organizacional:** El equipo necesita tiempo para volverse competente y eficiente en el uso de las herramientas y la metodología.
- **Reutilización de Activos:** Uno de los mayores retornos proviene de la reutilización de modelos y librerías de componentes. Desarrollar una librería de componentes de sistema (ej. modelos de sensores, actuadores, algoritmos de control) desde cero puede requerir un esfuerzo considerable (una estimación de 9-10 meses para componentes complejos puede ser razonable). Sin embargo, una vez que esta librería está creada, validada y gestionada (curada), el desarrollo de nuevos sistemas o variantes que utilicen esos componentes se acelera drásticamente. Lo que antes requería meses de desarrollo desde cero, puede reducirse a semanas o un mes de adaptación e integración, aprovechando los activos de modelado existentes.
- *La cuantificación de los beneficios "invisibles"*: Uno de los mayores desafíos reside en comunicar el valor de los "costes evitados" (es decir, errores que no llegaron a materializarse gracias a la detección temprana en el modelo). Estos ahorros, aunque reales, son difíciles de medir y presentar en métricas de retorno de inversión a corto plazo que la dirección suele exigir.

Esta desconexión temporal entre la inversión y el retorno tangible crea un riesgo significativo de que las organizaciones, especialmente aquellas bajo presión de resultados a corto plazo, perciban MBSE como un gasto excesivo y decidan abandonar el esfuerzo antes de que los beneficios a largo plazo puedan realizarse plenamente. La falta de resultados visibles inmediatos puede llevar a la pérdida de apoyo directivo y a la reversión a métodos tradicionales, perdiendo así la inversión inicial realizada.

La adopción exitosa de MBSE requiere, por tanto, una visión estratégica a largo plazo y un compromiso sostenido por parte de la dirección de la empresa. Es fundamental entender que MBSE es una inversión en la capacidad de ingeniería de la organización, cuyos frutos más importantes, como la eficiencia radical obtenida mediante la reutilización de activos de modelado, se cosechan una vez superada la fase inicial de construcción de modelos y adaptación cultural. La clave reside en gestionar las expectativas, comunicar el valor a largo plazo y asegurar la financiación y el apoyo necesarios para atravesar la fase inicial de inversión hasta alcanzar la fase de beneficios exponenciales derivados de un ecosistema de modelos maduros y reutilizables [Accenture and Navantia, sf].

Esta dicotomía entre la inversión inicial y el retorno a largo plazo evidencia que la adopción del MBSE no puede ser improvisada. Por el contrario, requiere de un plan estratégico bien definido, como se detalla en la siguiente hoja de ruta para su implementación organizacional.

4.3 Hoja de ruta para la adopción de MBSE

Una cosa es desarrollar un proyecto puntual sobre un sistema de interés simplificado en el ámbito de la investigación con un equipo reducido. Otra muy distinta es plantear la aplicación de esta tecnología a nivel organizacional, dentro de un grupo de trabajo en el que colaboran decenas o incluso centenas de ingenieros y especialistas. El MBSE no es solo una herramienta, es una transformación de la cultura de trabajo. A continuación, se expone una hoja de ruta detallada para guiar la implementación del MBSE en una organización compleja [Visure Solutions, sf].

Fase 1: Planificación y Alineación

Paso 1: Evaluación de Desafíos

Objetivo: Comprender en profundidad los desafíos actuales que enfrenta la empresa, tanto por las condiciones del mercado como por la competencia. Es crucial identificar los puntos débiles en los procesos de ingeniería y desarrollo actuales.
Participantes Clave: Representantes de Clientes, Analistas de Mercado, Equipos de Inteligencia Competitiva.

Paso 2: Alineación de las Partes Interesadas

Objetivo: Asegurar que todos los involucrados clave entiendan los beneficios y el alcance del MBSE. Se busca obtener su compromiso a través de sesiones de formación y reuniones iniciales.
Participantes Clave: Equipo de Proyecto, Ejecutivos, Usuarios Finales.

Paso 3: Definir Metas

Objetivo: Articular de manera clara los objetivos de la implementación y el cumplimiento normativo. Estas metas deben ser medibles y realistas.
Participantes Clave: Gerentes de Proyecto, Oficiales de Cumplimiento, Arquitectos de Sistemas.

Paso 4: Identificar KPIs

Objetivo: Establecer los Indicadores Clave de Rendimiento (KPIs) que permitirán medir la efectividad de la implementación de MBSE. Estos indicadores deben estar alineados con los objetivos definidos en el paso anterior (p. ej., reducción del tiempo de diseño, disminución de errores, etc.).
Participantes Clave: Gerentes de Proyecto, Equipos de Garantía de Calidad.

Fase 2: Implementación y Revisión

Paso 5: Proyecto Piloto

Objetivo: Lanzar un proyecto de MBSE a pequeña escala para probar la metodología y realizar la implementación inicial. Este piloto sirve como un campo de pruebas controlado para validar las herramientas y procesos.
Acciones Clave:

- Implementar colaboración en tiempo real para modelos compartidos, abordando las interdependencias entre subsistemas.
- Conectar con departamentos posteriores a la ingeniería para compartir datos de forma continua y obtener retroalimentación.

Participantes Clave: Equipo de Proyecto, Ingenieros de Sistemas, Especialistas de TI.

Paso 6: Puntos de Resolución y Revisión de Métricas **Objetivo:** Discutir los beneficios observados y abordar los desafíos o bloqueos encontrados durante el proyecto piloto. Se revisan los KPIs para evaluar el progreso.
Acciones Clave: Durante esta fase es fundamental demostrar el valor del MBSE:

- **Identificar Problemas Temprano:** Mostrar cómo la detección temprana de problemas contribuye a una mayor fiabilidad.
- **Asegurar la Trazabilidad:** Enfatizar el valor de entender el impacto de los cambios en la fiabilidad del sistema.
- **Promover la Adaptabilidad:** Discutir cómo la adaptabilidad del MBSE acelera los plazos y la agilidad del proyecto.

Participantes Clave: Gerentes de Proyecto, Ingenieros de Sistemas, Representantes de las Partes Interesadas.

Fase 3: Expansión y Futuro

Paso 8: Pasos Futuros

Objetivo: Explorar extensiones potenciales y la aplicación de MBSE a tecnologías más nuevas o áreas más complejas de la organización.
Participantes Clave: Expertos de la Industria, Especialistas en Tecnología, Equipos de Planificación Estratégica.

4.3.1 Componentes y Prácticas Clave para el Éxito

Para que la adopción de MBSE sea exitosa, es fundamental integrar los siguientes componentes y prácticas a lo largo de todo el proceso:

Componentes Clave

- **Integración de Datos Exhaustiva:** Asegurar que los modelos puedan integrar datos de diversas fuentes para crear una visión completa y unificada del sistema.
- **Optimización Avanzada del Diseño:** Utilizar los modelos para explorar y optimizar diferentes alternativas de diseño de manera eficiente.
- **Validación Rigurosa:** Aplicar procesos de validación y verificación rigurosos sobre los modelos para asegurar su corrección y precisión.

Mejores Prácticas Continuas

- **Fomentar la Reutilización del Diseño:** Utilizar el hilo digital (digital thread) de MBSE para habilitar la adaptabilidad y la reutilización de componentes de diseño en futuros productos.
- **Actualizar los Modelos Regularmente:** Mantener los modelos vivos mediante la simulación y las pruebas de datos para reducir las fallas por falta de actualización.
- **Gestionar los Modelos:** Asegurar la coherencia, la colaboración, la trazabilidad, el control de cambios y la mitigación efectiva de riesgos en todos los modelos del sistema.

La aplicación de MBSE es un proceso iterativo que transforma la manera en que una empresa aborda la ingeniería, llevando a una mayor eficiencia, calidad e innovación.

5 Conclusiones y Líneas Futuras

5.1 Síntesis de las Conclusiones Principales (Aprendizajes técnicos y profesionales)

En esta sección se consolidan tanto las competencias en herramientas específicas como las habilidades conceptuales de ingeniería de sistemas desarrolladas a lo largo del proyecto.

A. Nivel de Programas y Herramientas

A nivel de herramientas y procesos, se ha alcanzado un dominio integral del ecosistema MBSE. El modelado arquitectónico se ha llevado a cabo íntegramente en Capella, aplicando la metodología ARCADIA en todas sus capas. La capacidad de la herramienta se ha extendido mediante el uso de add-ons clave como Python4Capella para la automatización de flujos de trabajo, PVMT para la gestión de parámetros de diseño y Requirements Viewpoint para la integración de la normativa. Este modelo arquitectónico ha servido como *fuente única de verdad* para la Verificación y Validación (V&V), conectándose con MATLAB para ejecutar análisis de fiabilidad (FTA) y con Simulink para simular el rendimiento dinámico del sistema (ECS). La gestión de requisitos se ha anclado en herramientas como MATLAB Requirement Toolbox, asegurando la trazabilidad desde la normativa hasta el modelo. Finalmente, el uso de la IA generativa (Gemini) ha demostrado ser un acelerador eficaz para tareas de conceptualización y estructuración de datos. En conjunto, este flujo de trabajo simula de forma parcial la operación de un entorno PLM moderno.

B. Nivel Conceptual

A nivel conceptual y de ingeniería, este proyecto ha sido fundamental para desarrollar un *pensamiento sistémico* avanzado, abordando el diseño desde una perspectiva holística. Se ha adquirido una habilidad crítica para la *descomposición funcional y lógica* de problemas complejos, así como un profundo entendimiento de la *trazabilidad* como pilar para la gestión del cambio. La integración de simulaciones ha permitido aplicar de forma práctica el análisis de tipo *trade-off*, evaluando cuantitativamente el impacto de las decisiones de diseño sobre el rendimiento, la seguridad y el coste. Este enfoque de V&V continua es clave para mitigar riesgos. Adicionalmente, se ha obtenido un conocimiento profundo tanto de la normativa aeronáutica (CS-25) como de la arquitectura específica del sistema ATA 21.

5.2 Cumplimiento de los Objetivos del TFG

El presente Trabajo de Fin de Grado (TFG) se estructuró en torno a una serie de objetivos específicos, cuyo nivel de cumplimiento se detalla a continuación:

1. **Investigación y comprensión de la metodología MBSE y el framework ARCADIA:** Este objetivo se cumplió satisfactoriamente. Se llevó a cabo una investigación exhaustiva sobre MBSE, abarcando sus principios, beneficios y desafíos. Se profundizó en el framework ARCADIA, aplicándolo de forma práctica al caso de estudio a través de sus cuatro capas arquitectónicas (Operacional, Sistema, Lógica y Física). La comprensión demostrada se refleja en la estructura del trabajo y en la elaboración de los diagramas clave en Capella.
2. **Aplicación del framework ARCADIA al diseño de un subsistema de aeronave (Sistema de Aire Acondicionado y Presurización - ATA 21):** Este objetivo central se alcanzó con éxito. El Sistema de Aire Acondicionado y Presurización se modeló rigurosamente siguiendo las fases de ARCADIA. Se definieron los actores operacionales, sus necesidades y capacidades; se identificaron las funciones e interacciones del sistema; se descompuso lógicamente y se estableció una arquitectura física con redundancias. La creación de los diagramas OAB, SAB, LAB y PAB es la prueba tangible de este cumplimiento.
3. **Utilización de la herramienta Capella para la creación y gestión de modelos de sistema:** Se logró un dominio de Capella como herramienta principal de modelado. Se generaron todos los diagramas y elementos arquitectónicos necesarios en las diferentes capas de ARCADIA. Además, se exploró y aplicó el uso de add-ons como Python4Capella y PVMT, lo que demuestra un manejo avanzado de la herramienta y la capacidad de adaptarla a necesidades específicas del proyecto.
4. **Integración del modelo de Capella con herramientas de simulación (MATLAB/Simulink) para la validación y verificación de requisitos de diseño:** Este objetivo crucial se cumplió eficazmente. Se estableció una integración exitosa entre Capella y MATLAB/Simulink mediante scripts de Python4Capella. Se demostró cómo los parámetros de diseño del modelo de Capella pueden transferirse a un modelo de Simulink para simular el comportamiento del sistema y, posteriormente, cómo los resultados de esta simulación pueden utilizarse para verificar el cumplimiento de requisitos críticos (ej. CS-25.841.a.1 sobre altitud de cabina).
5. **Desarrollo de un modelo de Análisis de Árbol de Fallos en MATLAB para evaluar la fiabilidad del sistema y justificar redundancias:** Este objetivo se alcanzó mediante la creación de un modelo FTA en MATLAB. Dicho modelo no solo calculó la probabilidad de fallo catastrófico de la presurización de cabina, sino que también justificó la necesidad de redundancias (PACKs, OFV, CPCs) conforme a la normativa CS-25.1309. Además, el modelo incorpora un análisis de compromiso (trade-off) entre coste y fiabilidad, añadiendo valor a la evaluación de la arquitectura.
6. **Demostración del valor añadido de MBSE frente a enfoques tradicionales basados en documentos:** A lo largo del trabajo, se reflexionó sobre las ventajas de MBSE. En las conclusiones, se contrastan los beneficios de la coherencia del modelo, la trazabilidad intrínseca y la capacidad de integrar análisis tempranos (como las simulaciones), frente a las limitaciones de los enfoques basados en documentos, donde la información tiende a estar aislada y los errores se detectan tardíamente, incrementando los costes de corrección. La experiencia práctica con Capella y la integración con MATLAB refuerzan esta demostración.
7. **Identificación de limitaciones y propuesta de líneas futuras de investigación:** Se reconocieron y documentaron las limitaciones inherentes a un TFG de este alcance, como la imposibilidad de realizar un estudio de mercado detallado de componentes o la simplificación de ciertos escenarios. Asimismo, se plantearon diversas líneas futuras de trabajo que podrían expandir y profundizar la aplicación de MBSE y los análisis realizados, demostrando una visión crítica y prospectiva.

5.3 Limitaciones del Trabajo Realizado

El presente trabajo de investigación ha presentado las siguientes limitaciones:

- **Ausencia de un estudio de mercado exhaustivo de componentes:** No se llevó a cabo una selección y dimensionamiento de componentes físicos basada en datos reales de rendimiento, coste o disponibilidad de mercado.
- **Simplificación de escenarios de simulación:** Los modelos de simulación se utilizaron como "caja negra" con un nivel de detalle predefinido, sin profundizar en su creación desde cero o en la exploración de un amplio rango de escenarios operacionales complejos.
- **Familiarización inicial con add-ons:** La curva de aprendizaje con ciertas herramientas o add-ons (como PVMT, Python4Capella, Requirements Viewpoint,...) pudo haber limitado la implementación de todas las funcionalidades deseadas, como la asociación directa de valores de parámetros a los requisitos.
- **Modelo FTA simplificado:** El modelo de Análisis de Árbol de Fallos (FTA) se basó en hipótesis de independencia de fallos y tasas de fallo constantes, simplificaciones en comparación con los análisis de fiabilidad de la industria real. No se realizó una simulación de Monte Carlo completa debido a los recursos computacionales. Tampoco llegó a realizarse un estudio/cálculo de las probabilidades singulares de los componentes.
- **Tiempo y recursos** Como trabajo académico, el proyecto tuvo un alcance limitado en tiempo y recursos, lo que impidió la profundización en todas las áreas de la ingeniería de sistemas (ej., verificación y validación con pruebas físicas, gestión de la configuración avanzada, optimización de diseño con múltiples objetivos, MDAO,...).
- **Implementación parcial de requisitos normativos:** Aunque se incorporaron requisitos clave del CS-25, no se llevó a cabo un estudio exhaustivo de toda la normativa aplicable ni su completa integración en el modelo.

5.3.1 Diversidad Limitada de Requisitos

La diversidad de requisitos a los que se aplicó el modelo MBSE fue limitada. En un proyecto de ingeniería de sistemas ideal, no solo se gestionan requisitos normativos (como los CS-25), sino también:

Requisitos de misión Provenientes de los objetivos y el propósito general del sistema (por ejemplo, para un avión, "transportar X pasajeros a Y distancia con Z eficiencia").

Requisitos de la Solicitud de Propuesta (RFP) Específicos de un proceso de licitación, donde el cliente detalla sus necesidades y expectativas a los proveedores.

Requisitos de económicos y de empresa Derivados directamente de las necesidades y expectativas del usuario final o de la empresa encargada del desarrollo.

Requisitos de experiencias anteriores en sistemas similares Derivados de lecciones aprendidas, éxitos y fracasos en proyectos similares. Su objetivo es evitar repetir errores y capitalizar el conocimiento adquirido.

Requisitos de los Interesados (Stakeholder Requirements) Engloban las necesidades, expectativas y restricciones de todas las partes involucradas (cliente, usuarios finales, operadores, personal de mantenimiento, inversores, etc.). Los requisitos económicos son clave en todo problema ingenieril.

La ausencia de estos tipos de requisitos en el estudio se debe a la naturaleza académica del proyecto, que no se enmarcó en un contexto empresarial real con clientes y procesos de licitación definidos. Esto restringió la aplicación de MBSE a un conjunto de requisitos predominantemente técnicos y de cumplimiento normativo, sin explorar la complejidad de la gestión de requisitos de negocio y operativos presentes en un proyecto industrial.

5.4 Recomendaciones y Posibles Líneas de Trabajo Futuro

A continuación, se presentan varias líneas de trabajo futuro que podrían expandir y profundizar el análisis realizado, tanto desde una perspectiva de investigación como de aplicación industrial.

5.4.1 Profundización del Modelo MBSE y Análisis del Sistema

- **Integración de Parámetros Cuantificables en Requisitos:** Explorar y desarrollar métodos para asociar valores de parámetros (umbrales, rangos, especificaciones de rendimiento) directamente a los requisitos en herramientas de gestión (*DOORS*, *MATLAB Requirement Toolbox*) o mediante Viewpoints personalizados en *Capella*. El objetivo es que estos valores sean parte integral del requisito para facilitar su trazabilidad, validación automática y verificación.
- **Desarrollo de Diagramas de Modo y Estados para el Sistema de Aire Acondicionado:** Incorporar modelos de comportamiento basados en estados y modos operativos para el sistema ACPC. La creación de *State Machines* o *State-Transition Diagrams* enriquecería el modelo con una dimensión temporal y reactiva crucial para la validación de sistemas críticos en diferentes fases de vuelo o ante situaciones anómalas.
- **Profundización en el Uso de Interfaces en el Sistema de Aire Acondicionado:** Realizar un estudio detallado de las interfaces del sistema (ACPC). Esto implicaría modelar con mayor granularidad las interfaces eléctricas, neumáticas y de datos entre el **ATA 21** y otros sistemas de la aeronave para identificar puntos de fallo, cuellos de botella o requisitos de compatibilidad no explorados.

5.4.2 Expansión de Capacidades de Simulación y V&V

- **Implementación de Más Tipos de Simulaciones y Conexión con CAD:** Plantear un futuro proceso de diseño donde el modelo MBSE se integre con más tipos de simulación y con software CAD (*CATIA, SolidWorks*).
  - **Simulaciones de Rendimiento Aerotérmico (CFD):** Utilizar herramientas como *Ansys Fluent* u *OpenFOAM* para validar requisitos de caudal, uniformidad de temperatura y eficiencia de intercambiadores.
  - **Simulaciones de Consumo Energético y Eficiencia:** Evaluar el consumo de potencia del ACPC en diferentes fases de vuelo con herramientas como *Simscape Electrical*.
  - **Simulaciones de Mantenibilidad y Diagnóstico (PHM):** Desarrollar modelos en *MATLAB/Simulink* que simulen la degradación de componentes para diseñar estrategias de mantenimiento predictivo.

5.4.3 Desarrollo de Herramientas y Metodologías Avanzadas

- **Automatización Conceptual mediante IA Generativa (Gemini, ChatGPT):**
  - *Generación de Arquitecturas a partir de Texto:* Investigar la viabilidad de entrenar modelos de lenguaje para generar una conceptualización inicial de la arquitectura (cadenas funcionales, componentes, interfaces) a partir de requisitos textuales, e incluso traducir esta arquitectura redactada a scripts de *Python4Capella* que se han capaces de dibujar los modelos directamente en Capella.

  - *Asistencia en la Redacción de Modelos:* Utilizar la IA como asistente para la redacción de descripciones detalladas de elementos del modelo, asegurando la coherencia y guiando al ingeniero.

- **Desarrollo de un Plugin RAMS (Fiabilidad, Disponibilidad, Mantenibilidad y Seguridad) en Capella:** Investigar y prototipar un add-on para *Capella* que facilite la realización de análisis RAMS directamente en el modelo. Esto podría incluir funcionalidades para la gestión de **FMECA**, integración con cálculos de fiabilidad y generación automática de informes.
- **Aplicación de MBSE al Diseño de Satélites:** Extender la metodología MBSE y el análisis de compensaciones coste-fiabilidad al diseño de sistemas satelitales, donde la optimización simultánea de estos dos factores bajo severas restricciones es crítica.

Apéndice A
Script para Requisitos en MATLAB

Este anexo presenta el script completo desarrollado en MATLAB para la importación y estructuración automática de los requisitos de la normativa EASA CS-25 en el entorno de Simulink, a través de la Requirements Toolbox. El código está diseñado para ser modular y mantenible, creando una jerarquía clara que refleja la estructura de la normativa.

Código A.1 Script de generación de requisitos EASA CS-25 en MATLAB.

```
1
2  % ========================================================================
3  % ===   SCRIPT DE GENERACI?N DE REQUISITOS - ESTRUCTURA EASA CS-25    ===
4  % ========================================================================
5  % VERSI?N REFACTORIZADA: C?digo estructurado con un patr?n homog?neo para
6  % mayor claridad y mantenibilidad. La l?gica y el orden de los requisitos
7  % originales se mantienen intactos.
8  %
9  % MODIFICACI?N: Se ha a?adido el ID de cada requisito al inicio de su
10 % descripci?n de texto y se han corregido los caracteres de espacio no
11 % v?lidos (non-ASCII).
12 % ========================================================================
13
14 %% --- CONFIGURACI?N INICIAL Y ATRIBUTOS ---
15
16 % Limpieza del entorno (opcional, pero recomendado)
17 clear;
18 clc;
19
20 % Definici?n del archivo y creaci?n del conjunto de requisitos
21 reqSetFileName = 'EASA_CS25_Requirements_Structured_ID.slreqx';
22 myReqSet = slreq.new(reqSetFileName);
23 disp(['Creado nuevo conjunto de requisitos: ', myReqSet.Name]);
24
25 % Definici?n de atributos personalizados para el conjunto
26 myReqSet.addAttribute('Severity', 'Combobox', 'List', {'Unset', 'Catastrophic', '
       Hazardous', 'Major'});
27 myReqSet.addAttribute('Source', 'Combobox', 'List', {'Unset', 'CS-25', 'AMC-25', '
       Project'});
28 myReqSet.addAttribute('RequirementType', 'Combobox', 'List', {'Unset', 'Safety', '
       Design', 'Interface', 'Test', 'Performance', 'Compliance'});
29 disp('Atributos personalizados definidos.');
30
31
32 %% --- ESTRUCTURA JER?RQUICA PRINCIPAL ---
33
```

```
% --- RA?Z PRINCIPAL ---
reqEASARoot = myReqSet.add('Text', '[EASA-CS25] Normativa EASA CS-25 para Sistemas de Aire Acondicionado y Presurizaci?n', 'Id', 'EASA-CS25');
reqEASARoot.Description = 'Conjunto de requisitos que cubre el dise?o, la implementaci?n y la certificaci?n de un sistema de control ambiental (ECS).';
disp(['Creado requisito ra?z: ', reqEASARoot.Id]);

% --- LIBRO 1 y LIBRO 2 ---
reqLibro1 = reqEASARoot.add('Text', '[EASA-CS25.CS] Libro 1: Especificaciones de Certificaci?n (CS)', 'Id', 'EASA-CS25.CS');
reqLibro1.Description = 'Cuerpo normativo esencial que rige el dise?o y la operaci?n de las aeronaves.';

reqLibro2 = reqEASARoot.add('Text', '[EASA-CS25.AMC] Libro 2: Medios Aceptables de Cumplimiento (AMC)', 'Id', 'EASA-CS25.AMC');
reqLibro2.Description = 'Directrices y metodolog?as validadas por la EASA para demostrar la conformidad con las especificaciones del Libro 1.';
disp('Creada estructura de Libro 1 (EASA-CS25.CS) y Libro 2 (EASA-CS25.AMC).');


%% --- POBLANDO LIBRO 1: ESPECIFICACIONES DE CERTIFICACI?N (CS) ---
disp('--- Poblando Libro 1: Especificaciones de Certificaci?n (CS) ---');

% --- Creaci?n de la Subparte D ---
reqSubparteD = reqLibro1.add('Text', '[CS-BOOK1.D] Subparte D - Dise?o y Construcci?n', 'Id', 'CS-BOOK1.D');

% --- Requisito: CS 25.1309 Equipos, Sistemas e Instalaciones ---
req1309 = reqSubparteD.add('Text', '[CS-25.1309] CS 25.1309: Equipos, Sistemas e Instalaciones', 'Id', 'CS-25.1309');
req1309.setAttribute('Source', 'CS-25');
    % Sub-requisitos de CS 25.1309
    req1309_a = req1309.add('Text', '[CS-25.1309.a] Funcionamiento Previsto: Los equipos deben funcionar seg?n lo previsto bajo todas las condiciones operativas.', 'Id', 'CS-25.1309.a');
    req1309_a.setAttribute('RequirementType', 'Design');
    req1309_b = req1309.add('Text', '[CS-25.1309.b] Seguridad de Otros Equipos: Los equipos no deben ser un peligro ni afectar adversamente a otros sistemas.', 'Id', 'CS-25.1309.b');
    req1309_b.setAttribute('RequirementType', 'Safety');
    reqGestionFallos = req1309.add('Text', '[CS-25.1309.c] Gesti?n de Fallos: Los sistemas deben dise?arse para mitigar el efecto de los fallos.', 'Id', 'CS-25.1309.c');
    reqGestionFallos.setAttribute('RequirementType', 'Safety');
        % Desglose de Gesti?n de Fallos
        reqFalloCatastrofico = reqGestionFallos.add('Text', '[CS-25.1309.c.1] Cualquier fallo catastr?fico debe ser extremadamente improbable y no resultar de un fallo ?nico.', 'Id', 'CS-25.1309.c.1');
        reqFalloCatastrofico.setAttribute('Severity', 'Catastrophic');
        reqFalloPeligroso = reqGestionFallos.add('Text', '[CS-25.1309.c.2] Cualquier fallo peligroso debe ser extremadamente remoto.', 'Id', 'CS-25.1309.c.2');
        reqFalloPeligroso.setAttribute('Severity', 'Hazardous');
        reqFalloMayor = reqGestionFallos.add('Text', '[CS-25.1309.c.3] Cualquier fallo mayor debe ser remoto.', 'Id', 'CS-25.1309.c.3');
        reqFalloMayor.setAttribute('Severity', 'Major');
    req1309_d = req1309.add('Text', '[CS-25.1309.d] Informaci?n a la Tripulaci?n: Se debe proporcionar informaci?n sobre condiciones inseguras y alertas para acci?n correctiva.', 'Id', 'CS-25.1309.d');
    req1309_d.setAttribute('RequirementType', 'Interface');
    req1309_e = req1309.add('Text', '[CS-25.1309.e] Minimizaci?n de Errores: El dise?o de controles e indicaciones debe minimizar los errores de la tripulaci?n.', 'Id', 'CS-25.1309.e');
```

```
74      req1309_e.setAttribute('RequirementType', 'Interface');
75
76  % --- Requisito: CS 25.1438 Sistemas de Presurizaci?n y Neum?ticos ---
77  req1438 = reqSubparteD.add('Text', '[CS-25.1438] CS 25.1438: Sistemas de Presurizaci?n
        y Neum?ticos de Baja Presi?n', 'Id', 'CS-25.1438');
78  req1438.setAttribute('Source', 'CS-25');
79      % Sub-requisitos de CS 25.1438
80      req1438_a = req1438.add('Text', '[CS-25.1438.a] Cumplimiento General: Deben cumplir
         con CS 25.1309, considerando roturas o fugas excesivas.', 'Id', 'CS-25.1438.a');
81      req1438_a.setAttribute('RequirementType', 'Compliance');
82
83  % --- Requisito: CS 25.831 Ventilaci?n ---
84  req831 = reqSubparteD.add('Text', '[CS-25.831] CS 25.831: Ventilaci?n', 'Id', 'CS
        -25.831');
85  req831.setAttribute('Source', 'CS-25');
86      % Sub-requisitos de CS 25.831
87      req831_a = req831.add('Text', '[CS-25.831.a] Suministro de Aire: Cada compartimento
         debe ser ventilado (Tripulaci?n >= 0.28 m?/min).', 'Id', 'CS-25.831.a');
88      req831_a.setAttribute('RequirementType', 'Design');
89      req831_b_quality = req831.add('Text', '[CS-25.831.b] Calidad del Aire: El aire debe
         estar libre de concentraciones da?inas.', 'Id', 'CS-25.831.b');
90      req831_b_quality.setAttribute('RequirementType', 'Design');
91          % Desglose de Calidad del Aire
92          req831_b1_co = req831_b_quality.add('Text', '[CS-25.831.b.1] L?mite de Mon?xido
         de Carbono (CO): No superior a 1 parte en 20,000.', 'Id', 'CS-25.831.b.1');
93          req831_b2_co2 = req831_b_quality.add('Text', '[CS-25.831.b.2] L?mite de Di?xido
         de Carbono (CO2): No debe exceder 0.5% por volumen.', 'Id', 'CS-25.831.b.2');
94      req831_c = req831.add('Text', '[CS-25.831.c] Condiciones Post-Fallo: La calidad del
         aire debe mantenerse tras fallos razonablemente probables.', 'Id', 'CS-25.831.c');
95      req831_c.setAttribute('RequirementType', 'Safety');
96      req831_d = req831.add('Text', '[CS-25.831.d] Evacuaci?n de Humo: Debe ser posible
        evacuar r?pidamente el humo de la cabina de vuelo.', 'Id', 'CS-25.831.d');
97      req831_d.setAttribute('RequirementType', 'Safety');
98      req831_e = req831.add('Text', '[CS-25.831.e] Control Independiente: Tripulaci?n
        debe poder controlar su ventilaci?n independientemente.', 'Id', 'CS-25.831.e');
99      req831_e.setAttribute('RequirementType', 'Interface');
100
101 % --- Requisito: CS 25.841 Cabinas Presurizadas ---
102 req841 = reqSubparteD.add('Text', '[CS-25.841] CS 25.841: Cabinas Presurizadas', 'Id', '
        CS-25.841');
103 req841.setAttribute('Source', 'CS-25');
104     % Sub-requisitos de CS 25.841
105     req841_a_alt = req841.add('Text', '[CS-25.841.a] Niveles de Altitud de Cabina', 'Id'
        , 'CS-25.841.a');
106         % Desglose de Niveles de Altitud
107         req841_a1 = req841_a_alt.add('Text', '[CS-25.841.a.1] Altitud de Cabina (Normal)
        : No superior a 8,000 ft en operaci?n normal.', 'Id', 'CS-25.841.a.1');
108         req841_a1.setAttribute('RequirementType', 'Design');
109         req841_a2 = req841_a_alt.add('Text', '[CS-25.841.a.2] Altitud de Cabina (Fallo):
         No superior a 15,000 ft tras un fallo probable del sistema.', 'Id', 'CS-25.841.a.2
        ');
110         req841_a2.setAttribute('RequirementType', 'Safety');
111     req841_b_equip = req841.add('Text', '[CS-25.841.b] V?lvulas, Controles e
        Indicadores', 'Id', 'CS-25.841.b');
112     req841_b_equip.setAttribute('RequirementType', 'Design');
113         % Desglose de Equipamiento
114         req841_b1 = req841_b_equip.add('Text', '[CS-25.841.b.1] V?lvulas de Alivio de
        Presi?n Positiva (Dos unidades).', 'Id', 'CS-25.841.b.1');
115         req841_b2 = req841_b_equip.add('Text', '[CS-25.841.b.2] V?lvulas de Alivio de
        Presi?n Negativa (Dos unidades o equivalente).', 'Id', 'CS-25.841.b.2');
116         req841_b3 = req841_b_equip.add('Text', '[CS-25.841.b.3] Medio de Ecualizaci?n R
        ?pida de presi?n.', 'Id', 'CS-25.841.b.3');
```

```
117         req841_b4 = req841_b_equip.add('Text', '[CS-25.841.b.4] Regulador Autom?tico o
        Manual para controlar el flujo.', 'Id', 'CS-25.841.b.4');
118         req841_b5 = req841_b_equip.add('Text', '[CS-25.841.b.5] Instrumentaci?n:
        Indicadores de P diferencial, altitud cabina y vari?metro cabina.', 'Id', 'CS
        -25.841.b.5');
119         req841_b5.setAttribute('RequirementType', 'Interface');
120         req841_b6 = req841_b_equip.add('Text', '[CS-25.841.b.6] Indicaciones de
        Advertencia: Alerta si la altitud de cabina > 10,000 ft.', 'Id', 'CS-25.841.b.6');
121         req841_b6.setAttribute('RequirementType', 'Interface');
122         req841_b7 = req841_b_equip.add('Text', '[CS-25.841.b.7] Placa de Advertencia:
        Si la estructura no soporta P diferencial m?ximo en aterrizaje.', 'Id', 'CS-25.841.
        b.7');
123         req841_b8 = req841_b_equip.add('Text', '[CS-25.841.b.8] Sensores de Presi?n:
        Ubicaci?n y dise?o deben asegurar activaci?n sin demoras.', 'Id', 'CS-25.841.b.8');
124
125     % --- Requisito: CS 25.832 Concentraci?n de Ozono en Cabina ---
126     req832 = reqSubparteD.add('Text', '[CS-25.832] CS 25.832: Concentraci?n de Ozono en
            Cabina', 'Id', 'CS-25.832');
127     req832.setAttribute('Source', 'CS-25');
128         % Sub-requisitos de CS 25.832
129         req832_a1 = req832.add('Text', '[CS-25.832.a.1] L?mite de Ozono 1: No exceder 0.25
            ppm por encima de FL320.', 'Id', 'CS-25.832.a.1');
130         req832_a1.setAttribute('RequirementType', 'Design');
131         req832_a2 = req832.add('Text', '[CS-25.832.a.2] L?mite de Ozono 2: No exceder 0.1
            ppm (promedio 3h) por encima de FL270.', 'Id', 'CS-25.832.a.2');
132         req832_a2.setAttribute('RequirementType', 'Design');
133
134     % --- Requisito: CS 25.1041 Requisitos de Rendimiento T?rmico ---
135     req1041 = reqSubparteD.add('Text', '[CS-25.1041] CS 25.1041: Requisitos de Rendimiento
            T?rmico', 'Id', 'CS-25.1041');
136     req1041.setAttribute('Source', 'CS-25');
137     req1041.setAttribute('RequirementType', 'Performance');
138         % Sub-requisitos de CS 25.1041
139         req1041_1 = req1041.add('Text', '[CS-25.1041.1] Rendimiento de Enfriamiento en
            Tierra: Enfriar cabina de 47?C a 21?C en menos de 30 min.', 'Id', 'CS-25.1041.1');
140         req1041_2 = req1041.add('Text', '[CS-25.1041.2] Rendimiento de Calentamiento en
            Tierra: Calentar cabina de -40?C a 24?C en menos de 30 min.', 'Id', 'CS-25.1041.2')
            ;
141         req1041_3 = req1041.add('Text', '[CS-25.1041.3] Rendimiento de Enfriamiento en
            Crucero: Aplicable a vuelos supers?nicos con alto incremento de T por velocidad.',
            'Id', 'CS-25.1041.3');
142
143     % --- Requisito: CS 25.1091 Toma de Aire (Air Intake) ---
144     req1091 = reqSubparteD.add('Text', '[CS-25.1091] CS 25.1091: Toma de Aire (Air Intake)',
             'Id', 'CS-25.1091');
145     req1091.setAttribute('Source', 'CS-25');
146         % Sub-requisitos de CS 25.1091
147         req1091_a = req1091.add('Text', '[CS-25.1091.a] Suministro de Aire al Motor: Debe
            suministrar el aire requerido por el motor en todas las condiciones.', 'Id', 'CS
            -25.1091.a');
148         req1091_a.setAttribute('RequirementType', 'Design');
149
150         % Requisito de ubicaci?n de la toma (NUEVO)
151         req1091_c = req1091.add('Text', '[CS-25.1091.c] Ubicaci?n de las Tomas: Las tomas
            de aire no deben abrirse dentro del carenado del motor, a menos que est? aislado
            por un diafragma ign?fugo.', 'Id', 'CS-25.1091.c');
152         req1091_c.setAttribute('RequirementType', 'Design');
153
154         req1091_d = req1091.add('Text', '[CS-25.1091.d] Prevenci?n de Ingesti?n: Prevenir
            la ingesti?n de cantidades peligrosas de fluidos, agua o FOD.', 'Id', 'CS-25.1091.d
            ');
155         req1091_d.setAttribute('RequirementType', 'Safety');
```

```
156
157     % Requisito de resistencia a ingesti?n de objetos (NUEVO)
158     req1091_e = req1091.add('Text', '[CS-25.1091.e] Resistencia a FOD: El dise?o de la
        toma debe poder soportar la ingesti?n de objetos extra?os sin fallos que creen un
        peligro.', 'Id', 'CS-25.1091.e');
159     req1091_e.setAttribute('RequirementType', 'Safety');
160
161
162
163 %% --- POBLANDO LIBRO 2: MEDIOS ACEPTABLES DE CUMPLIMIENTO (AMC) ---
164 disp('--- Poblando Libro 2: Medios Aceptables de Cumplimiento (AMC) ---');
165
166 % --- AMC: AMC a CS 25.1438 Pruebas de Sistemas Neum?ticos ---
167 amc1438 = reqLibro2.add('Text', '[AMC-25.1438] AMC a CS 25.1438: Pruebas de Sistemas
        Neum?ticos', 'Id', 'AMC-25.1438');
168 amc1438.setAttribute('Source', 'AMC-25');
169     % Sub-requisitos de AMC 25.1438
170     amc1438_1_res = amc1438.add('Text', '[AMC-25.1438.1] Resistencia de Componentes', '
        Id', 'AMC-25.1438.1');
171     amc1438_1_res.setAttribute('RequirementType', 'Test');
172         % Desglose de Resistencia de Componentes
173         amc1438_1_1 = amc1438_1_res.add('Text', '[AMC-25.1438.1.1] Un elemento cr?tico
        debe funcionar sin distorsi?n tras someterse a "Condiciones 1".', 'Id', 'AMC
        -25.1438.1.1');
174         amc1438_1_2 = amc1438_1_res.add('Text', '[AMC-25.1438.1.2] Un elemento cr?tico
        debe soportar "Condiciones 2" sin reventar.', 'Id', 'AMC-25.1438.1.2');
175         amc1438_1_3 = amc1438_1_res.add('Text', '[AMC-25.1438.1.3] El sistema debe
        resistir la manipulaci?n probable durante el mantenimiento.', 'Id', 'AMC
        -25.1438.1.3');
176     amc1438_2_pru = amc1438.add('Text', '[AMC-25.1438.2] Pruebas de Componentes', 'Id',
         'AMC-25.1438.2');
177     amc1438_2_pru.setAttribute('RequirementType', 'Test');
178         % Desglose de Pruebas de Componentes
179         amc1438_2_1 = amc1438_2_pru.add('Text', '[AMC-25.1438.2.1] Pruebas Est?ticas:
        Probar est?ticamente cada elemento cr?tico.', 'Id', 'AMC-25.1438.2.1');
180         amc1438_2_2 = amc1438_2_pru.add('Text', '[AMC-25.1438.2.2] Pruebas de Fatiga:
        Someter a pruebas de fatiga los elementos cuya falla sea peligrosa.', 'Id', 'AMC
        -25.1438.2.2');
181
182 % --- AMC: AMC a CS 25.841 (CS 25.843) Pruebas para Cabinas Presurizadas ---
183 amc843 = reqLibro2.add('Text', '[AMC-25.843] AMC a CS 25.841 (CS 25.843): Pruebas para
        Cabinas Presurizadas', 'Id', 'AMC-25.843');
184 amc843.setAttribute('Source', 'AMC-25');
185     % Sub-requisitos de AMC 25.843
186     amc843_a = amc843.add('Text', '[AMC-25.843.a] Prueba de Resistencia Estructural: La
         cabina debe ser probada como un recipiente a presi?n.', 'Id', 'AMC-25.843.a');
187     amc843_a.setAttribute('RequirementType', 'Test');
188     amc843_b_func = amc843.add('Text', '[AMC-25.843.b] Pruebas Funcionales', 'Id', 'AMC
        -25.843.b');
189     amc843_b_func.setAttribute('RequirementType', 'Test');
190         % Desglose de Pruebas Funcionales
191         amc843_b1 = amc843_b_func.add('Text', '[AMC-25.843.b.1] Probar el
        funcionamiento y capacidad de las v?lvulas de alivio y descarga.', 'Id', 'AMC
        -25.843.b.1');
192         amc843_b2 = amc843_b_func.add('Text', '[AMC-25.843.b.2] Probar el sistema de
        presurizaci?n bajo todas las condiciones (P, T, humedad).', 'Id', 'AMC-25.843.b.2')
        ;
193         amc843_b3 = amc843_b_func.add('Text', '[AMC-25.843.b.3] Realizar pruebas en
        vuelo para verificar el rendimiento en ascensos y descensos.', 'Id', 'AMC-25.843.b
        .3');
```

```
194         amc843_b4 = amc843_b_func.add('Text', '[AMC-25.843.b.4] Probar que las puertas
        y salidas de emergencia operan correctamente post-pruebas de vuelo.', 'Id', 'AMC
        -25.843.b.4');
195
196 % --- AMC: AMC a CS 25.832 Demostraci?n de Cumplimiento de L?mites de Ozono ---
197 amc832 = reqLibro2.add('Text', '[AMC-25.832] AMC a CS 25.832: Demostraci?n de
        Cumplimiento de L?mites de Ozono', 'Id', 'AMC-25.832');
198 amc832.setAttribute('Source', 'AMC-25');
199     % Sub-requisitos de AMC 25.832
200     amc832_1 = amc832.add('Text', '[AMC-25.832.1] Demostrar mediante an?lisis o pruebas
         que se mantendr?n las concentraciones de ozono por debajo de los l?mites.', 'Id',
        'AMC-25.832.1');
201     amc832_1.setAttribute('RequirementType', 'Compliance');
202
203     % Requisito de dise?o del catalizador (NUEVO)
204     amc832_2_design = amc832.add('Text', '[PROJ-25.832.2] Dise?o del Catalizador: El
        sistema debe incluir un convertidor catal?tico que asegure una reducci?n de al
        menos el 80% del ozono entrante.', 'Id', 'PROJ-25.832.2');
205     amc832_2_design.setAttribute('Source', 'Project');
206     amc832_2_design.setAttribute('RequirementType', 'Design');
207
208
209 % --- AMC: AMC para Pruebas de Enfriamiento (CS 25.1043 & 25.1045) ---
210 amcCooling = reqLibro2.add('Text', '[AMC-25.104X] AMC para Pruebas de Enfriamiento (CS
        25.1043 & 25.1045)', 'Id', 'AMC-25.104X');
211 amcCooling.setAttribute('Source', 'AMC-25');
212     % Sub-requisitos de AMC Cooling
213     amc1043 = amcCooling.add('Text', '[AMC-25.1043] Condiciones de Prueba (CS 25.1043)',
         'Id', 'AMC-25.1043');
214     amc1043.setAttribute('RequirementType', 'Test');
215         % Desglose de Condiciones de Prueba
216         amc1043_a = amc1043.add('Text', '[AMC-25.1043.a] General: Cumplimiento debe
        demostrarse en condiciones cr?ticas. Temp. deben corregirse si se difiere de la m?
        xima.', 'Id', 'AMC-25.1043.a');
217         amc1043_b = amc1043.add('Text', '[AMC-25.1043.b] Temperatura Ambiente M?xima:
        Al menos 37.8?C a nivel del mar, con un lapso de -6.6?C/km.', 'Id', 'AMC-25.1043.b'
        );
218         amc1043_c = amc1043.add('Text', '[AMC-25.1043.c] Factor de Correcci?n: Corregir
         temperaturas sumando la diferencia entre T? amb. m?xima y T? amb. de la prueba.',
        'Id', 'AMC-25.1043.c');
219     amc1045 = amcCooling.add('Text', '[AMC-25.1045] Procedimientos de Prueba (CS
        25.1045)', 'Id', 'AMC-25.1045');
220     amc1045.setAttribute('RequirementType', 'Test');
221         % Desglose de Procedimientos de Prueba
222         amc1045_a = amc1045.add('Text', '[AMC-25.1045.a] Fases de Vuelo: Demostrar
        cumplimiento en despegue, ascenso, crucero y aterrizaje en configuraci?n cr?tica.',
         'Id', 'AMC-25.1045.a');
223         amc1045_b = amc1045.add('Text', '[AMC-25.1045.b] Estabilizaci?n: T?
        estabilizada si cambio < 1?C/min. Estabilizar antes de cada fase de vuelo.', 'Id',
        'AMC-25.1045.b');
224         amc1045_c = amc1045.add('Text', '[AMC-25.1045.c] Duraci?n: Continuar pruebas
        hasta estabilizaci?n, fin de fase de vuelo o alcanzar un l?mite operativo.', 'Id',
        'AMC-25.1045.c');
225
226
227 %% --- GUARDADO Y FINALIZACI?N ---
228
229 % Guardado final del archivo .slreqx
230 myReqSet.save();
231 disp('------------------------------------------------------');
232 disp(['Conjunto de requisitos completo y estructurado guardado en: ', fullfile(pwd,
        reqSetFileName)]);
```

```
233 disp('Proceso finalizado.');
234
235 % Opcional: Abrir el editor de requisitos para visualizar el resultado
236 % slreq.editor;
237 % slreq.open(reqSetFileName);
```

Apéndice B
Script de Conexión Capella y MATLAB/Simulink

Este anexo detalla el script de Python utilizado dentro del entorno de Capella, a través de la extensión Python4Capella. Su función principal es establecer una comunicación bidireccional con un modelo del Sistema de Control Ambiental (ECS) desarrollado en MATLAB/Simulink. El script extrae parámetros de diseño del modelo de Capella, los inyecta en la simulación de Simulink, ejecuta el análisis y recupera los resultados para validar los requisitos arquitectónicos, automatizando así parte del proceso de verificación.

Código B.1 Script en Python para la conexión de Capella con el modelo en MatLAB/Simulink..

```
# -*- coding: utf-8 -*-
# include needed for the Capella API
include('workspace://Python4Capella/simplified_api/capella.py')
if False:
    from simplified_api.capella import *
# include needed for the PVMT API
include('workspace://Python4Capella/simplified_api/pvmt.py')
if False:
    from simplified_api.pvmt import *
# include needed for utilities
include('workspace://Python4Capella/utilities/CapellaPlatform.py')
if False:
    from utilities.CapellaPlatform import *
# include needed for the requirement API
include('workspace://Python4Capella/simplified_api/requirement.py')
if False:
    from simplified_api.requirement import *

import matlab.engine
import os # Recommended for handling paths

def set_p_v_value(elem, PVName, value):
    # Itera sobre los grupos de propiedades que posee el objeto Java subyacente del elemento.
    for group in elem.get_java_object().getOwnedPropertyValueGroups():
        # Dentro de cada grupo, itera sobre los valores de las propiedades.
        for pv in group.getOwnedPropertyValues():
            # Comprueba si el nombre de la propiedad actual coincide con el nombre buscado.
            if PVName == pv.getName():
```

```
                # Si hay coincidencia, establece el nuevo valor.
                pv.setValue(value)
                # Termina la ejecución de la función para mayor eficiencia.
                return

# --- Configuration ---
# It's better to define paths at the beginning for easier management.
# Consider using relative paths if the script is in a project structure.
MATLAB_PROJECT_PATH = r'C:\Users\migue\Documents\MATLAB\Examples\R2024b\simscape\
    AircraftEnvironmentalControlSystemExample'
AIRD_PATH = '/Sistema/Sistema.aird'

# --- Helper Function for MATLAB Simulation ---
def run_matlab_simulation(matlab_engine, T_bleed, p_bleed, ACM_flow_rate, T_turbine,
    Cabin_T, Cabin_min_p, Numero_de_pasajeros, Calor_emitido_pasajero_tiempo,
    Humedad_generada_pasajero, Temperatura_exhalacion, Calor_equipos,
    Coef_transferencia_calor_fuselaje, Volumen_cabina, Puerto_A, Puerto_B, Puerto_C,
    Ram_flow_rate):
    """
    Runs a specific MATLAB simulation case and returns the results.
    """
    try:
        # Set variables in MATLAB workspace
        matlab_engine.workspace['T_bleed'] = T_bleed
        matlab_engine.workspace['p_bleed'] = p_bleed
        matlab_engine.workspace['ACM_flow_rate'] = ACM_flow_rate
        matlab_engine.workspace['T_turbine'] = T_turbine
        matlab_engine.workspace['Cabin_T'] = Cabin_T
        matlab_engine.workspace['Cabin_min_p'] = Cabin_min_p
        matlab_engine.workspace['Numero_de_pasajeros'] = Numero_de_pasajeros
        matlab_engine.workspace['Calor_emitido_pasajero_tiempo'] =
    Calor_emitido_pasajero_tiempo
        matlab_engine.workspace['Humedad_generada_pasajero'] =
    Humedad_generada_pasajero
        matlab_engine.workspace['Temperatura_exhalacion'] = Temperatura_exhalacion
        matlab_engine.workspace['Calor_equipos'] = Calor_equipos
        matlab_engine.workspace['Coef_transferencia_calor_fuselaje'] =
    Coef_transferencia_calor_fuselaje
        matlab_engine.workspace['Volumen_cabina'] = Volumen_cabina
        matlab_engine.workspace['Puerto_A'] = Puerto_A
        matlab_engine.workspace['Puerto_B'] = Puerto_B
        matlab_engine.workspace['Puerto_C'] = Puerto_C
        matlab_engine.workspace['Ram_flow_rate'] = Ram_flow_rate

        # Define script paths
        os.path.join(MATLAB_PROJECT_PATH)

        # Run MATLAB scripts
        eng.run(r"C:\Users\migue\Documents\MATLAB\Examples\R2024b\simscape\
    AircraftEnvironmentalControlSystemExample\
    AircraftEnvironmentalControlSystemPlot1MDot.m", nargout=0)

        # Retrieve results from MATLAB workspace
        presion_cabina = eng.workspace["presion_cabina_min"]
        print(f'  Presión mínima de la cabina = {presion_cabina:.2f} kPa,')
        return presion_cabina

    except Exception as e:
        print(f"Error running MATLAB simulation")
        return 0

# --- Main Script ---
```

```
print('Script execution started.')

# Start MATLAB Engine
try:
    eng = matlab.engine.start_matlab()
except Exception as e:
    print(f"Error starting MATLAB engine: {e}")
    # Exit if MATLAB can't be started
    exit()

# Initialize Capella Model
model = CapellaModel()
model.open(AIRD_PATH)
se = model.get_system_engineering()

# Get all Logical Components and Property Value names
all_logical_actors = se.get_all_contents_by_type(LogicalActor)
all_logical_components = se.get_all_contents_by_type(LogicalComponent)
all_pv_names = []
for lc in all_logical_components:
    for pv_name in PVMT.get_p_v_names(lc):
        if pv_name not in all_pv_names:
            all_pv_names.append(pv_name)
for la in all_logical_actors:
    for pv_name in PVMT.get_p_v_names(la):
        if pv_name not in all_pv_names:
            all_pv_names.append(pv_name)

# Process each Logical Component
for lc in all_logical_components:
    print(f"Processing Component: {lc.get_name()}")
    for pv_name in PVMT.get_p_v_names(lc):
        pv_value = PVMT.get_p_v_value(lc, pv_name)
        if str(pv_value) != 'None':
            if pv_name == 'T_bleed':
                T_bleed = float(pv_value)
                print(f'  {pv_name}: {T_bleed} K')
            elif pv_name == 'p_bleed':
                p_bleed = float(pv_value)
                print(f'  {pv_name}: {p_bleed} kPa')
            elif pv_name == 'ACM_flow_rate':
                ACM_flow_rate = float(pv_value)
                print(f'  {pv_name}: {ACM_flow_rate} kg/s')
            elif pv_name == 'T_turbine':
                T_turbine = float(pv_value) + 273.15
                print(f'  {pv_name}: {T_turbine:.2f} K')
            elif pv_name == 'Cabin_T':
                Cabin_T = float(pv_value) + 273.15
                print(f'  {pv_name}: {Cabin_T:.2f} K')
            elif pv_name == 'Cabin_min_p':
                Cabin_min_p = float(pv_value)
                print(f'  {pv_name}: {Cabin_min_p} kPa')
            elif pv_name == 'Numero_de_pasajeros':
                Numero_de_pasajeros = float(pv_value)
                print(f'  {pv_name}: {Numero_de_pasajeros}')
            elif pv_name == 'Calor_emitido_pasajero_tiempo':
                Calor_emitido_pasajero_tiempo = float(pv_value)
                print(f'  {pv_name}: {Calor_emitido_pasajero_tiempo} W/pasajero')
            elif pv_name == 'Humedad_generada_pasajero':
                Humedad_generada_pasajero = float(pv_value)
                print(f'  {pv_name}: {Humedad_generada_pasajero} kg_agua/s/pasajero')
            elif pv_name == 'Temperatura_exhalacion':
```

```
                Temperatura_exhalacion = float(pv_value) + 273.15
                print(f'  {pv_name}: {Temperatura_exhalacion:.2f} K')
            elif pv_name == 'Calor_equipos':
                Calor_equipos = float(pv_value)
                print(f'  {pv_name}: {Calor_equipos} W')
            elif pv_name == 'Coef_transferencia_calor_fuselaje':
                Coef_transferencia_calor_fuselaje = float(pv_value)
                print(f'  {pv_name}: {Coef_transferencia_calor_fuselaje} W/(m^2 K)')
            elif pv_name == 'Volumen_cabina':
                Volumen_cabina = float(pv_value)
                print(f'  {pv_name}: {Volumen_cabina} m^3')
            elif pv_name == 'Puerto_A':
                Puerto_A = float(pv_value)
                print(f'  {pv_name}: {Puerto_A} m^2')
            elif pv_name == 'Puerto_B':
                Puerto_B = float(pv_value)
                print(f'  {pv_name}: {Puerto_B} m^2')
            elif pv_name == 'Puerto_C':
                Puerto_C = float(pv_value)
                print(f'  {pv_name}: {Puerto_C} m^2')
            elif pv_name == 'Ram_flow_rate':
                Ram_flow_rate = float(pv_value)
                print(f'  {pv_name}: {Ram_flow_rate} kg/s')

for la in all_logical_actors:
    print(f"Processing Actors: {la.get_name()}")
    for pv_name in PVMT.get_p_v_names(la):
        pv_value = PVMT.get_p_v_value(la, pv_name)
        if str(pv_value) != 'None':
            if pv_name == 'Presion_8000m':
                presion_requerida = float(pv_value)
                print(f'  {pv_name}: {presion_requerida} Pa')

# Run simulation
presion_actual = run_matlab_simulation(eng, T_bleed, p_bleed, ACM_flow_rate, T_turbine,
     Cabin_T, Cabin_min_p, Numero_de_pasajeros, Calor_emitido_pasajero_tiempo,
    Humedad_generada_pasajero, Temperatura_exhalacion, Calor_equipos,
    Coef_transferencia_calor_fuselaje, Volumen_cabina, Puerto_A, Puerto_B, Puerto_C,
    Ram_flow_rate)
print(presion_actual)

model.start_transaction()
try:
    for req in se.get_all_contents_by_type(Requirement):
        print(f"Evaluando Requisito ID: {req.get_id()} - Nombre: {req.get_long_name()}")


        if req.get_long_name() == 'CS-25.841.a.1':
            print("  -> Requisito de presurización encontrado. Verificando condición...
    ")

            if presion_actual >= presion_requerida:
                req.set_prefix('REQUISITO SATISFECHO')
                print(f"  -> RESULTADO: SATISFECHO (Presión actual: {presion_actual},
    Requerida: {presion_requerida})")
            else:
                req.set_prefix('REQUISITO NO SATISFECHO')
                print(f"  -> RESULTADO: NO SATISFECHO (Presión actual: {presion_actual},
     Requerida: {presion_requerida})")
except:
    # if something went wrong we rollback the transaction
    model.rollback_transaction()
```

```
197         raise
198     else:
199         # if everything is ok we commit the transaction
200         model.commit_transaction()
201
202     eng.quit()
```

Apéndice C
Script de Conexión entre Capella y el modelo FTA en MATLAB

Este anexo contiene el código fuente del script de Python desarrollado para automatizar el flujo de trabajo entre el modelo arquitectónico en Capella y el script de análisis de fiabilidad en MATLAB. El script extrae los parámetros de diseño del modelo, ejecuta la simulación y verifica el resultado comparando con los requisitos definidos en Capella, actualizando su estado de satisfacción.

Código C.1 Script de Python para la extracción de parámetros, ejecución de simulación y verificación de requisitos..

```python
# -*- coding: utf-8 -*-
"""
Este script integra un modelo de Capella con un análisis de seguridad en MATLAB.
Su flujo de trabajo es el siguiente:
1. Inicia una conexión con un modelo de Capella y un motor de MATLAB.
2. Extrae parámetros cuantitativos (Property Values) de los componentes y actores
    lógicos del modelo de Capella.
3. Pasa estos parámetros a un script de MATLAB para ejecutar una simulación
    (presumiblemente un Análisis de Árbol de Fallos - FTA).
4. Recupera el resultado del análisis (probabilidad del evento tope).
5. Verifica si el resultado satisface un requisito específico en el modelo Capella.
6. Actualiza el estado del requisito en el modelo para reflejar si se ha
    satisfecho o no.
7. Cierra la conexión con el motor de MATLAB.
"""

# --- Inclusión de APIs de Python4Capella ---
# Esta sintaxis es específica del entorno de scripting de Capella y carga las APIs
    necesarias.
# El bloque "if False" es una técnica para ayudar a los IDEs con el autocompletado y
# el análisis estático sin ejecutar la importación estándar de Python.

# API principal de Capella
include('workspace://Python4Capella/simplified_api/capella.py')
if False:
    from simplified_api.capella import *

# API para la gestión de Property Values (PVMT)
include('workspace://Python4Capella/simplified_api/pvmt.py')
if False:
    from simplified_api.pvmt import *

```

```python
# Utilidades de la plataforma Capella
include('workspace://Python4Capella/utilities/CapellaPlatform.py')
if False:
    from utilities.CapellaPlatform import *

# API para la gestión de Requisitos
include('workspace://Python4Capella/simplified_api/requirement.py')
if False:
    from simplified_api.requirement import *

# --- Importaciones Estándar de Python ---
import matlab.engine
import os  # Recomendado para manejar rutas de archivos de forma robusta

# --- Definición de Rutas y Constantes ---
MATLAB_PROJECT_PATH = r'C:\Users\migue\Documents\MATLAB\Examples\R2024b\RAMS'
AIRD_PATH = '/Sistema/Sistema.aird'


# --- Funciones Auxiliares ---

def set_p_v_value(elem, pv_name, value):
    """
    Establece el valor de un Property Value (PV) específico para un elemento del modelo.

    Nota: Esta función está definida pero no se utiliza en el flujo principal de este script.

    Args:
        elem (EObject): El elemento de Capella cuyo PV se va a modificar.
        pv_name (str): El nombre del Property Value a buscar.
        value (any): El nuevo valor a establecer para el PV.
    """
    # Itera sobre los grupos de propiedades que posee el objeto Java subyacente del elemento.
    for group in elem.get_java_object().getOwnedPropertyValueGroups():
        # Dentro de cada grupo, itera sobre los valores de las propiedades.
        for pv in group.getOwnedPropertyValues():
            # Comprueba si el nombre de la propiedad actual coincide con el nombre buscado.
            if pv_name == pv.getName():
                # Si hay coincidencia, establece el nuevo valor.
                pv.setValue(str(value))
                # Termina la ejecución de la función para mayor eficiencia.
                return


def run_matlab_simulation(matlab_engine, P_Fallo_ControladorPack, P_Fallo_ACM,
    P_Fallo_FCV, Costo_ACM, Costo_FCV, Costo_ControladorPACK, N_Packs,
    P_Fallo_Motor_OFV, Costo_Motor_OFV, N_OFV, Costo_CPC, N_CPC, P_Fallo_CPC,
    P_Fallo_Estructural):
    """
    Ejecuta la simulación en MATLAB y devuelve los resultados.

    Args:
        matlab_engine: La instancia del motor de MATLAB.
        (Múltiples): Todos los parámetros extraídos de Capella necesarios para la simulación.

    Returns:
        float: La probabilidad del evento tope calculada, o 0 si ocurre un error.
    """
```

```
    try:
        # Asignar variables al workspace de MATLAB
        matlab_engine.workspace['P_Fallo_ControladorPack'] = P_Fallo_ControladorPack
        matlab_engine.workspace['P_Fallo_ACM'] = P_Fallo_ACM
        matlab_engine.workspace['P_Fallo_FCV'] = P_Fallo_FCV
        matlab_engine.workspace['Costo_ACM'] = Costo_ACM
        matlab_engine.workspace['Costo_FCV'] = Costo_FCV
        matlab_engine.workspace['Costo_ControladorPACK'] = Costo_ControladorPACK
        matlab_engine.workspace['N_Packs'] = N_Packs
        matlab_engine.workspace['P_Fallo_Motor_OFV'] = P_Fallo_Motor_OFV
        matlab_engine.workspace['Costo_Motor_OFV'] = Costo_Motor_OFV
        matlab_engine.workspace['N_OFV'] = N_OFV
        matlab_engine.workspace['P_Fallo_CPC'] = P_Fallo_CPC
        matlab_engine.workspace['Costo_CPC'] = Costo_CPC
        matlab_engine.workspace['N_CPC'] = N_CPC
        matlab_engine.workspace['P_Fallo_Estructural'] = P_Fallo_Estructural

        # Ejecutar el script de MATLAB
        # Se usa nargout=0 para indicar que no se esperan salidas directas de la funció
    n run.
        # Los resultados se recuperarán del workspace.
        matlab_script_path = os.path.join(MATLAB_PROJECT_PATH, "FTA_Aire_acondicionado.
    m")
        eng.run(matlab_script_path, nargout=0)

        # Recuperar resultados del workspace de MATLAB
        p_top_event_calculada = eng.workspace["P_Top_Event_Calculada"]

        print('Simulación realizada con éxito.')
        print('A continuación se mostrarán los parámetros output calculados:')
        print(f'P_Top_Event_Calculada: {p_top_event_calculada}')

        return p_top_event_calculada

    except Exception as e:
        print(f"Error durante la ejecución de la simulación en MATLAB: {e}")
        return 0


# --- Script Principal ---
print('Inicio de la ejecución del script.')

# 1. Iniciar el motor de MATLAB
try:
    print("Iniciando motor de MATLAB...")
    eng = matlab.engine.start_matlab()
    print("Motor de MATLAB iniciado correctamente.")
except Exception as e:
    print(f"Error crítico: No se pudo iniciar el motor de MATLAB: {e}")
    exit()

# 2. Inicializar el modelo de Capella
print(f"Abriendo modelo de Capella desde: {AIRD_PATH}")
model = CapellaModel()
model.open(AIRD_PATH)
se = model.get_system_engineering()
print("Modelo de Capella cargado.")

# 3. Extraer parámetros (Property Values) del modelo
print("\n--- Extrayendo parámetros de los Componentes y Actores Lógicos ---")
all_logical_actors = se.get_all_contents_by_type(LogicalActor)
all_logical_components = se.get_all_contents_by_type(LogicalComponent)
```

```
146
147 # Se combinan todos los elementos en una lista para iterar sobre ellos una sola vez.
148 all_elements = all_logical_components + all_logical_actors
149
150 # Se itera sobre cada elemento para buscar y asignar explícitamente cada Property Value.
151 for elem in all_elements:
152     # CORRECCIÓN: Se utiliza type(elem).__name__ en lugar de elem.eClass().getName()
153     # para obtener el nombre de la clase del objeto de una manera estándar en Python.
154     print(f"Procesando elemento: {elem.get_name()} (Tipo: {type(elem).__name__})")
155     for pv_name in PVMT.get_p_v_names(elem):
156         pv_value = PVMT.get_p_v_value(elem, pv_name)
157         if str(pv_value) != 'None':
158             # Estructura if/elif para asignar cada valor a su variable correspondiente.
159             # Este método es explícito pero menos flexible que usar un diccionario.
160             if pv_name == 'P_Fallo_ControladorPack':
161                 P_Fallo_ControladorPack = float(pv_value)
162                 print(f"  -> {pv_name}: {P_Fallo_ControladorPack}")
163             elif pv_name == 'P_Fallo_ACM':
164                 P_Fallo_ACM = float(pv_value)
165                 print(f"  -> {pv_name}: {P_Fallo_ACM}")
166             elif pv_name == 'P_Fallo_FCV':
167                 P_Fallo_FCV = float(pv_value)
168                 print(f"  -> {pv_name}: {P_Fallo_FCV}")
169             elif pv_name == 'Costo_ACM':
170                 Costo_ACM = float(pv_value)
171                 print(f"  -> {pv_name}: {Costo_ACM}")
172             elif pv_name == 'Costo_FCV':
173                 Costo_FCV = float(pv_value)
174                 print(f"  -> {pv_name}: {Costo_FCV}")
175             elif pv_name == 'Costo_ControladorPACK':
176                 Costo_ControladorPACK = float(pv_value)
177                 print(f"  -> {pv_name}: {Costo_ControladorPACK}")
178             elif pv_name == 'N_Packs':
179                 N_Packs = float(pv_value)
180                 print(f"  -> {pv_name}: {N_Packs}")
181             elif pv_name == 'P_Fallo_Motor_OFV':
182                 P_Fallo_Motor_OFV = float(pv_value)
183                 print(f"  -> {pv_name}: {P_Fallo_Motor_OFV}")
184             elif pv_name == 'Costo_Motor_OFV':
185                 Costo_Motor_OFV = float(pv_value)
186                 print(f"  -> {pv_name}: {Costo_Motor_OFV}")
187             elif pv_name == 'N_OFV':
188                 N_OFV = float(pv_value)
189                 print(f"  -> {pv_name}: {N_OFV}")
190             elif pv_name == 'Costo_CPC':
191                 Costo_CPC = float(pv_value)
192                 print(f"  -> {pv_name}: {Costo_CPC}")
193             elif pv_name == 'N_CPC':
194                 N_CPC = float(pv_value)
195                 print(f"  -> {pv_name}: {N_CPC}")
196             elif pv_name == 'P_Fallo_CPC':
197                 P_Fallo_CPC = float(pv_value)
198                 print(f"  -> {pv_name}: {P_Fallo_CPC}")
199             elif pv_name == 'P_Fallo_Estructural':
200                 P_Fallo_Estructural = float(pv_value)
201                 print(f"  -> {pv_name}: {P_Fallo_Estructural}")
202             elif pv_name == 'P_catastrofico_min':
203                 P_catastrofico_min = float(pv_value)
204                 print(f"  -> {pv_name}: {P_catastrofico_min}")
205
206 print("\nExtracción de parámetros finalizada.")
```

```
207
208
209 # 4. Ejecutar la simulación en MATLAB
210 print("\n--- Iniciando simulación en MATLAB ---")
211 # CORRECCIÓN: Se utilizan los nombres de variable correctos extraídos del modelo.
212 # Originalmente se usaban "P_f_ofv" y "P_f_est", que no se definían.
213 # Los nombres correctos son "P_Fallo_Motor_OFV" y "P_Fallo_Estructural".
214 p_actual = run_matlab_simulation(eng,
215                                  P_Fallo_ControladorPack, P_Fallo_ACM, P_Fallo_FCV,
216                                  Costo_ACM, Costo_FCV, Costo_ControladorPACK, N_Packs,
217                                  P_Fallo_Motor_OFV, Costo_Motor_OFV, N_OFV,
218                                  Costo_CPC, N_CPC, P_Fallo_CPC,
219                                  P_Fallo_Estructural)
220
221 # 5. Actualizar el modelo de Capella con los resultados
222 print("\n--- Actualizando el modelo de Capella con los resultados ---")
223 # Se inicia una transacción. Todos los cambios dentro de este bloque se aplicarán
224 # de forma atómica (o todos o ninguno).
225 model.start_transaction()
226 try:
227     # Iterar sobre todos los requisitos del modelo
228     for req in se.get_all_contents_by_type(Requirement):
229         print(f"Evaluando Requisito ID: {req.get_id()} - Nombre: {req.get_long_name()}")
230
231         # Buscar el requisito específico a verificar
232         if req.get_long_name() == 'CS-25.1309.c.1':
233             print("  -> Requisito de probabilidad encontrado. Verificando condición...")
234
235             # Comparar la probabilidad simulada con el umbral definido en el modelo
236             if p_actual <= P_catastrofico_min:
237                 req.set_prefix('REQUISITO SATISFECHO')
238                 print(f"  -> RESULTADO: SATISFECHO (Probabilidad simulada: {p_actual:.2
    E}, Requerida <= {P_catastrofico_min:.2E})")
239             else:
240                 req.set_prefix('REQUISITO NO SATISFECHO')
241                 print(f"  -> RESULTADO: NO SATISFECHO (Probabilidad simulada: {p_actual
    :.2E}, Requerida <= {P_catastrofico_min:.2E})")
242
243 except Exception as e:
244     # Si algo sale mal, se revierten todos los cambios hechos en la transacción.
245     print(f"\nError durante la actualización del modelo: {e}. Revirtiendo transacción.")
246     model.rollback_transaction()
247     raise  # Relanza la excepción para no ocultar el error.
248 else:
249     # Si todo ha ido bien, se confirman los cambios en el modelo.
250     print("\nActualización del modelo finalizada. Confirmando transacción.")
251     model.commit_transaction()
252
253 # 6. Detener el motor de MATLAB
254 print("\nDeteniendo motor de MATLAB...")
255 eng.quit()
256 print("Script finalizado.")
```

Apéndice D
Código del Modelo FTA (Fault Tree Analysis) del Sistema

Este anexo detalla el script de Python utilizado dentro del entorno de Capella, a través de la extensión Python4Capella, para la conexión bidireccional con el script de MATLAB. El modelo paramétrico de MATLAB realiza un Análisis de Árbol de Fallos (FTA) para un sistema de presurización. A partir de las tasas de fallo de los componentes y el número de redundancias, calcula analíticamente la probabilidad de fallo del sistema y genera una visualización dinámica del árbol. Además, ejecuta un análisis de compromiso (trade-off) que compara el coste de adquisición frente a la fiabilidad para distintas arquitecturas. Su objetivo es permitir la optimización del diseño basándose en la seguridad y el coste.

Código D.1 Apéndice IV: Modelo Parametrizado de Análisis de Árbol de Fallos (FTA)..

```
%% MODELO PARAMETRIZADO DE ANÁLISIS DE ÁRBOL DE FALLOS (FTA)
% Autor: Gemini AI
% Fecha: 30 de junio de 2025
% Descripción: Versión parametrizada del modelo FTA. Permite configurar el
%              número de packs, OFVs y CPCs.

%% 0. CONFIGURACIÓN DEL MODELO
%--------------------------------------------------------------------------
% Aquí puedes definir la arquitectura del sistema. El resto del script se
% adaptará a estos valores.
%--------------------------------------------------------------------------
%N_Packs = 2;   % Número de Packs de Aire Acondicionado (ej. 2 para A320, 3 para A380)
%N_OFV = 2;     % Número de Válvulas de Salida (Outflow Valves)
%N_CPC = 2;     % Número de Controladores de Presión de Cabina

fprintf('--- CONFIGURACIÓN DEL SISTEMA ---\n');
fprintf('Modelo configurado para:\n');
fprintf('- %d Packs de Aire Acondicionado\n', N_Packs);
fprintf('- %d Válvulas de Salida (OFV)\n', N_OFV);
fprintf('- %d Controladores de Presión (CPC)\n\n', N_CPC);

%% 1. PARÁMETROS DE ENTRADA Y TASAS de FALLO
% (Sin cambios en esta sección)
%P_Fallo_ACM = 1.5e-5;
%P_Fallo_FCV = 0.8e-5;
%P_Fallo_ControladorPack = 0.5e-5;
%P_Fallo_Motor_OFV = 2.0e-6;
%P_Fallo_CPC = 1.0e-6;
```

```
P_Perdida_ADIRU = 0; %Se desprecia la Probabilidad de  Perdida de la ADIRU
%P_Fallo_Estructural = 1.0e-9;

%% 1b. PARÁMETROS DE COSTE (Valores ilustrativos en Euros)
% Coste de adquisición de cada componente
%Costo_ACM = 80000;
%Costo_FCV = 15000;
%Costo_ControladorPack = 25000;
%Costo_Motor_OFV = 12000;
%Costo_CPC = 20000;

% Coste de CONSECUENCIA por fallo (reparación, AOG, mano de obra, etc.)
Costo_Consecuencia_Fallo_Componente = 250000;

%% 2. CÁLCULO ANALÍTICO DE PROBABILIDAD (Bottom-Up)
%{
    MODIFICACIÓN: Los cálculos de fallo de sistemas redundantes ahora usan
    el operador de potencia (^) con los parámetros de la Sección 0.
%}
P_Fallo_Pack_Individual = P_Fallo_ACM + P_Fallo_FCV + P_Fallo_ControladorPack;
P_Perdida_Total_Packs = P_Fallo_Pack_Individual ^ double(N_Packs);
P_Fallo_Todas_OFV = P_Fallo_Motor_OFV ^ double(N_OFV);
P_Fallo_Todos_CPC = P_Fallo_CPC ^ double(N_CPC);
P_Comando_Erroneo_OFV = P_Fallo_Todos_CPC + P_Perdida_ADIRU;
P_Fallo_Control_Salida = P_Fallo_Todas_OFV + P_Comando_Erroneo_OFV;
P_Tasa_Salida_Excede_Entrada = P_Perdida_Total_Packs + P_Fallo_Control_Salida;
P_Top_Event_Calculada = P_Tasa_Salida_Excede_Entrada + P_Fallo_Estructural;

fprintf('--- ANÁLISIS CUANTITATIVO ---\n');
fprintf('Probabilidad de fallo de TODOS los Packs (AND): %.2e\n', P_Perdida_Total_Packs)
    ;
fprintf('PROBABILIDAD DEL TOP EVENT (Calculada) POR HORA: %.4e\n\n',
    P_Top_Event_Calculada);

%% 3. VISUALIZACIÓN DEL ÁRBOL DE FALLOS
%{
    MODIFICACIÓN (CORREGIDA FINAL): Se asignan nombres únicos a cada nodo
    de puerta lógica (ej. 'OR (Top Event)') para cumplir con el requisito
    de 'digraph' de que todos los nombres de nodo sean únicos.
%}
fprintf('--- VISUALIZACIÓN ---\n');
fprintf('Construyendo grafo del FTA dinámicamente...\n');

% Inicializar variables para el grafo
s = [];
t = [];
node_names = {};
node_colors = [];
node_idx = 0; % Contador para el índice del nodo actual

% Definir colores
color_top   = [1 0.6 0.6];     % Rojo
color_gate  = [0.9 0.9 0.9];   % Gris
color_inter = [0.6 0.8 1];     % Azul
color_basic = [1 1 0.7];     % Amarillo

% --- Construcción del Grafo (Paso a Paso) ---
% Nivel 0: Top Event
node_idx = node_idx + 1;
idx_top = node_idx;
node_names{idx_top} = 'TOP: Pérdida Presurización';
node_colors(idx_top, :) = color_top;
```

```
89
90  % Nivel 1
91  node_idx = node_idx + 1;
92  idx_or_top = node_idx;
93  node_names{idx_or_top} = 'OR (Top Event)';
94  node_colors(idx_or_top, :) = color_gate;
95  s(end+1) = idx_or_top; t(end+1) = idx_top; % Conexión
96
97  % Nivel 2
98  node_idx = node_idx + 1;
99  idx_salida_mayor_entrada = node_idx;
100 node_names{idx_salida_mayor_entrada} = 'Salida > Entrada';
101 node_colors(idx_salida_mayor_entrada, :) = color_inter;
102 s(end+1) = idx_salida_mayor_entrada; t(end+1) = idx_or_top; % Conexión
103
104 node_idx = node_idx + 1;
105 idx_fallo_estructural = node_idx;
106 node_names{idx_fallo_estructural} = 'Fallo Estructural';
107 node_colors(idx_fallo_estructural, :) = color_basic;
108 s(end+1) = idx_fallo_estructural; t(end+1) = idx_or_top; % Conexión
109
110 % Nivel 3
111 node_idx = node_idx + 1;
112 idx_or_salida = node_idx;
113 node_names{idx_or_salida} = 'OR (Causas Principales)'; % <- NOMBRE ÚNICO
114 node_colors(idx_or_salida, :) = color_gate;
115 s(end+1) = idx_or_salida; t(end+1) = idx_salida_mayor_entrada; % Conexión
116
117 % Nivel 4
118 node_idx = node_idx + 1;
119 idx_perdida_packs = node_idx;
120 node_names{idx_perdida_packs} = 'Pérdida Total Packs';
121 node_colors(idx_perdida_packs, :) = color_inter;
122 s(end+1) = idx_perdida_packs; t(end+1) = idx_or_salida; % Conexión
123
124 node_idx = node_idx + 1;
125 idx_fallo_control_salida = node_idx;
126 node_names{idx_fallo_control_salida} = 'Fallo Control Salida';
127 node_colors(idx_fallo_control_salida, :) = color_inter;
128 s(end+1) = idx_fallo_control_salida; t(end+1) = idx_or_salida; % Conexión
129
130 % Nivel 5: Packs (Parte Dinámica)
131 node_idx = node_idx + 1;
132 idx_and_packs = node_idx;
133 node_names{idx_and_packs} = 'AND (Fallo Packs)'; % <- NOMBRE ÚNICO
134 node_colors(idx_and_packs, :) = color_gate;
135 s(end+1) = idx_and_packs; t(end+1) = idx_perdida_packs; % Conexión
136
137 % Bucle para crear dinámicamente los nodos de fallo de cada Pack
138 for i = 1:N_Packs
139    node_idx = node_idx + 1;
140    idx_pack_i = node_idx;
141    node_names{idx_pack_i} = sprintf('Fallo Pack %d', i);
142    node_colors(idx_pack_i, :) = color_inter;
143    s(end+1) = idx_pack_i; t(end+1) = idx_and_packs; % Conexión
144 end
145
146 % --- Fin de la Construcción ---
147 % Visualizar el grafo final
148 G = digraph(s, t, [], node_names);
149 figure('Name', 'Visualización Parametrizada del FTA');
150 p = plot(G, 'Layout', 'layered', 'Direction', 'up');
```

```
p.NodeColor = node_colors;
p.EdgeColor = [0.4 0.4 0.4];
p.LineWidth = 1.5;
p.NodeLabel = G.Nodes.Name;
title(sprintf('Diagrama del FTA para %d Packs', N_Packs));
set(gca, 'XTick', [], 'YTick', []);
fprintf('Visualización generada.\n\n');

%% 4. SIMULACIÓN DE MONTE CARLO
%{
    MODIFICACIÓN: La simulación ahora crea una matriz para los fallos de
    los packs. Usa un bucle para simular cada pack y la función all() para
    verificar si TODOS los packs han fallado.
%}
fprintf('--- SIMULACIÓN DE MONTE CARLO ---\n');
num_simulaciones = 3e8; % 10 Millones, un valor razonable
fprintf('Ejecutando %.0e simulaciones...\n', num_simulaciones);

% Matriz booleana para los fallos de los packs (filas=simulaciones, cols=packs)
fallo_pack_sim = false(num_simulaciones, N_Packs);

for i = 1:N_Packs
   % Se necesitan 3 números aleatorios por pack
   rand_samples = rand(num_simulaciones, 3);
   fallo_acm = rand_samples(:,1) < P_Fallo_ACM;
   fallo_fcv = rand_samples(:,2) < P_Fallo_FCV;
   fallo_ctrl = rand_samples(:,3) < P_Fallo_ControladorPack;

   % Puerta OR para el fallo de un pack individual
   fallo_pack_sim(:, i) = fallo_acm | fallo_fcv | fallo_ctrl;
end

% Puerta AND para la pérdida total de packs.
% all(..., 2) comprueba si TODAS las columnas de cada fila son true.
perdida_total_packs_sim = all(fallo_pack_sim, 2);

% Top Event (simplificado, como antes)
fallo_estructural_sim = rand(num_simulaciones, 1) < P_Fallo_Estructural;
top_event_sim = perdida_total_packs_sim | fallo_estructural_sim;

num_fallos = sum(top_event_sim);
P_Top_Event_Simulada = num_fallos / num_simulaciones;

fprintf('Número de fallos del sistema detectados en la simulación: %d\n', num_fallos);
fprintf('PROBABILIDAD DEL TOP EVENT (Simulada) POR HORA: %.4e\n', P_Top_Event_Simulada);



%% 5. ANÁLISIS DE COSTE
fprintf('--- ANÁLISIS DE COSTE ---\n');

% --- 5.1. Coste Total de Adquisición (Bill of Materials) ---
Costo_Unitario_Pack = Costo_ACM + Costo_FCV + Costo_ControladorPack;
Costo_Total_Packs = N_Packs * Costo_Unitario_Pack;
Costo_Total_OFVs = N_OFV * Costo_Motor_OFV;
Costo_Total_CPCs = N_CPC * Costo_CPC;
Costo_Total_Adquisicion = Costo_Total_Packs + Costo_Total_OFVs + Costo_Total_CPCs;

fprintf('Coste de Adquisición de 1 Pack: %.2f euros\n', Costo_Unitario_Pack);
fprintf('Coste Total de Adquisición del Sistema: %.2f euros\n', Costo_Total_Adquisicion)
    ;

```

```
% --- 5.2. Coste Esperado de Fallo por Hora ---
E_Costo_Packs = N_Packs * P_Fallo_Pack_Individual * Costo_Consecuencia_Fallo_Componente;

E_Costo_OFVs = N_OFV * P_Fallo_Motor_OFV * Costo_Consecuencia_Fallo_Componente;
E_Costo_CPCs = N_CPC * P_Fallo_CPC * Costo_Consecuencia_Fallo_Componente;
E_Costo_Total_Fallo_por_Hora = E_Costo_Packs + E_Costo_OFVs + E_Costo_CPCs;

fprintf('Coste Esperado de Fallo por Hora (Mantenimiento por Fiabilidad): %.2f euros/
    hora\n\n', E_Costo_Total_Fallo_por_Hora);

% --- 5.3. Análisis de Compromiso Multidimensional (Trade-Off) ---
fprintf('--- ANÁLISIS DE COMPROMISO MULTIDIMENSIONAL ---\n');
fprintf('Calculando el espacio de diseño para Packs, OFVs y CPCs...\n');

% Definir los rangos para cada parámetro a analizar
rango_packs = 1:3;
rango_ofv = 1:2;
rango_cpc = 1:2;

% Se usará una tabla para almacenar los resultados de forma ordenada
% 1. Calcula el número total de filas que tendrá la tabla final.
num_filas = numel(rango_packs) * numel(rango_ofv) * numel(rango_cpc);
% 2. Define los nombres y tipos de las columnas (los mismos que ya usas).
varNames = {'Configuracion', 'N_Packs', 'N_OFV', 'N_CPC', 'Costo', 'Probabilidad'};
varTypes = {'string', 'double', 'double', 'double', 'double', 'double'};
% 3. Crea la tabla completa pero vacía.
resultados = table('Size', [num_filas, numel(varNames)], ...
                   'VariableTypes', varTypes, ...
                   'VariableNames', varNames);
idx = 1;

% Bucles anidados para iterar sobre cada combinación posible de la arquitectura
for n_p = rango_packs
   for n_o = rango_ofv
       for n_c = rango_cpc
           % --- Recalcular Coste y Fiabilidad para la configuración actual ---

           % Coste de adquisición para la configuración (n_p, n_o, n_c)
           costo_actual = (n_p * Costo_Unitario_Pack) + ...
                          (n_o * Costo_Motor_OFV) + ...
                          (n_c * Costo_CPC);

           % Probabilidad de fallo para la configuración (n_p, n_o, n_c)
           p_perdida_packs_n = P_Fallo_Pack_Individual ^ n_p;
           p_fallo_ofv_n = P_Fallo_Motor_OFV ^ n_o;
           p_fallo_cpc_n = P_Fallo_CPC ^ n_c;

           p_comando_erroneo_n = p_fallo_cpc_n; % Asumiendo que ADIRU no falla
           p_fallo_control_salida_n = p_fallo_ofv_n + p_comando_erroneo_n;
           p_tasa_salida_excede_entrada_n = p_perdida_packs_n +
    p_fallo_control_salida_n;
           prob_actual = p_tasa_salida_excede_entrada_n + P_Fallo_Estructural;

           % Almacenar los resultados
           resultados.Configuracion{idx} = sprintf('P%d,O%d,C%d', n_p, n_o, n_c);
           resultados.N_Packs(idx) = n_p;
           resultados.N_OFV(idx) = n_o;
           resultados.N_CPC(idx) = n_c;
           resultados.Costo(idx) = costo_actual;
           resultados.Probabilidad(idx) = prob_actual;
           idx = idx + 1;
       end
```

```
    end
end

disp('Espacio de diseño calculado:');
disp(resultados); % Muestra la tabla de resultados en la consola

% --- Generar el gráfico de Trade-Off Multidimensional ---
figure('Name', 'Análisis de Compromiso Multidimensional');
hold on;
grid on;

% Definir estilos para el plotting
colores = {'b', 'r', 'g', 'm'}; % Colores para N_Packs = 1, 2, 3, 4
marcadores = {'s', 'o'};       % Marcadores para N_OFV = 1, 2 (s=square, o=circle)

% Dibujar cada punto con el estilo que le corresponde
for i = 1:height(resultados)
    semilogy(resultados.Costo(i)/1e6, resultados.Probabilidad(i), ...
        'Marker', marcadores{resultados.N_OFV(i)}, ...
        'Color', colores{resultados.N_Packs(i)}, ...
        'MarkerSize', 10, ...
        'LineWidth', 2);

    % Añadir etiqueta de texto a cada punto
    text(resultados.Costo(i)/1e6, resultados.Probabilidad(i), ...
        ['  ' resultados.Configuracion{i}], ...
        'VerticalAlignment', 'middle', 'HorizontalAlignment', 'left', 'FontSize', 9);
end

% Configurar el gráfico
xlabel('Coste Total de Adquisición (Millones de euros)');
ylabel('Probabilidad de Fallo del Sistema por Hora (Escala Log)');
title('Compromiso (Trade-Off) entre Coste y Fiabilidad');
set(gca, 'YScale', 'log'); % Asegurar escala logarítmica

% Crear una leyenda manualmente para explicar los estilos
legend_handles = [];
legend_labels = {};

for i = 1:length(rango_packs)
    h = plot(NaN, NaN, '-', 'Color', colores{i}, 'LineWidth', 2); % Punto invisible
    legend_handles = [legend_handles, h];
    legend_labels{end+1} = sprintf('%d Pack(s)', rango_packs(i));
end

for i = 1:length(rango_ofv)
    h = plot(NaN, NaN, 'k', 'Marker', marcadores{i}, 'LineStyle', 'none'); % Punto
     invisible
    legend_handles = [legend_handles, h];
    legend_labels{end+1} = sprintf('%d OFV(s)', rango_ofv(i));
end

legend(legend_handles, legend_labels, 'Location', 'northeast');
hold off;
```

Bibliografía

[Accenture and Navantia, sf] Accenture and Navantia (s.f.). Documentación técnica interna. Material interno no recuperable públicamente.

[CIMdata, Inc., 2023] CIMdata, Inc. (2023). Cimdata plm market analysis report series. Technical report, CIMdata, Inc., Ann Arbor, MI. Los informes de CIMdata son una referencia estándar de la industria para el análisis de la cuota de mercado de PLM.

[CIMdata, Inc., 2024] CIMdata, Inc. (2024). Cimdata 2024 plm market analysis report series. Technical report, CIMdata. Informe de análisis de mercado que posiciona a Siemens como líder en el sector PLM.

[Dassault Systèmes, a] Dassault Systèmes. Magicgrid book of knowledge. Formulario de acceso al documento de conocimiento sobre la metodología MagicGrid®.

[Dassault Systèmes, b] Dassault Systèmes. Virtual twin experience.

[Dassault Systèmes, 2017] Dassault Systèmes (2017). Boeing to expand deployment of dassault systèmes' 3dexperience platform.

[Dassault Systèmes, 2021] Dassault Systèmes (2021). Customer stories: Driving the future of mobility. Se hace referencia a la adopción por parte de múltiples OEMs. Renault ha sido un cliente histórico y Tesla es conocido por su uso extensivo de las soluciones de DS.

[Dassault Systèmes, 2025] Dassault Systèmes (2025). Cameo systems modeler - product page.

[Delligatti, 2014] Delligatti, L. (2014). *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Addison-Wesley.

[Eclipse Foundation, 2025] Eclipse Foundation (2025). Eclipse capella - official website. *https://www.eclipse.org/capella/*. Página oficial del proyecto que confirma su naturaleza de código abierto, su gobernanza por la Fundación Eclipse y su origen en Thales.

[European Space Agency, 2022] European Space Agency (2022). MBSE Best Practices Project - Final Report Executive Summary. Technical report, European Space Agency (ESA). ESA Contract No. 4000133517/20/NL/CRS/is.

[European Space Agency, sf] European Space Agency (s.f.). Model based for system engineering (mb4se). *https://mb4se.esa.int/*.

[European Union Aviation Safety Agency, 2007] European Union Aviation Safety Agency (2007). Certification specifications and acceptable means of compliance for large aeroplanes cs-25. Technical report, European Union Aviation Safety Agency. Amendment 3.

[Hysko, 2023] Hysko, A. (2023). What should you model in mbse? LinkedIn.

[INCOSE, 2023] INCOSE (2023). *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. John Wiley and Sons, Inc., 5th edition.

[Martínez-Val and Pérez, sf] Martínez-Val, R. and Pérez, E. (s.f.). Aircraft environmental control system.

[Massachusetts Institute of Technology, sf] Massachusetts Institute of Technology (s.f.). Materiales de los cursos: Models in engineering, model-based systems engineering: Documentation and analysis, y quantitative methods in systems engineering.

[Object Management Group, 2022] Object Management Group (2022). *OMG Systems Modeling Language (OMG SysML) Specification*. Version 1.7.

[Peral González, 2018] Peral González, J. (2018). Dimensionado y proyecto básico de sistema de presurización y aire acondicionado de avión. Trabajo de fin de grado, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, Sevilla. Tutor: Diego Jerónimo Morillo Galeote.

[Roques, 2018] Roques, P. (2018). *Systems Architecture Modeling with the Arcadia Method: A Practical Guide to Capella*. ISTE Press - Elsevier.

[Siemens Digital Industries Software, 2021] Siemens Digital Industries Software (2021). Hyundai motor company and kia corporation select siemens for next-generation engineering and product data management. Anuncio oficial de la adopción de la plataforma de Siemens por un OEM líder del sector automotriz.

[Siemens Digital Industries Software, 2024] Siemens Digital Industries Software (2024). The comprehensive digital twin. *https://www.sw.siemens.com/en-US/digital-twin/*.

[Visure Solutions, sf] Visure Solutions (s.f.). Guía mbse: ¿qué es y cómo implementar la ingeniería de sistemas basada en modelos? *https://visuresolutions.com/es/mbse-guide/how-to-implement-mbse/*.

[Weigel, 2000] Weigel, A. L. (2000). An overview of the systems engineering knowledge domain assignment for esd.83: Research seminar in engineering systems. Technical report, Massachusetts Institute of Technology.

Acrónimos

ADIRU Air Data Inertial Reference Unit (Unidad de Referencia Inercial de Datos Aéreos).

AIMS Airplane Information Management System (Sistema de Gestión de Información del Avión).

APU Auxiliary Power Unit (Unidad de Potencia Auxiliar).

ARINC Aeronautical Radio, Incorporated (protocolo de comunicación para aviónica).

BDD Block Definition Diagram (Diagrama de Definición de Bloques).

BMC Bleed Monitoring Computer (Computador de Monitorización de Sangrado).

CPCS Cabin Pressure Control System (Sistema de Control de Presión de Cabina).

ECAM Electronic Centralized Aircraft Monitor (Monitor Centralizado Electrónico de la Aeronave).

ECS Environmental Control System (Sistema de Control Ambiental).

EICAS Engine Indicating and Crew Alerting System (Sistema de Indicación de Motores y Alerta a la Tripulación).

ESA European Space Agency (Agencia Espacial Europea).

FAA Federal Aviation Administration (Administración Federal de Aviación de EE. UU.).

FEA Finite Element Analysis (Análisis por Elementos Finitos).

FEM Finite Element Model (Modelo de Elementos Finitos).

FMS Flight Management System (Sistema de Gestión de Vuelo).

FOD Foreign Object Debris/Damage (Restos/Daños por Objetos Extraños).

FTA Fault Tree Analysis (Análisis de Árbol de Fallos).

HEPA High-Efficiency Particulate Air (Filtro de Aire de Alta Eficiencia).

HP High Pressure (Alta Presión).

ICD Interface Control Document (Documento de Control de Interfaces).

IP Intermediate Pressure (Presión Intermedia).

ISA International Standard Atmosphere (Atmósfera Estándar Internacional).

KPI Key Performance Indicator (Indicador Clave de Rendimiento).

LRU Line-Replaceable Unit (Unidad Reemplazable en Línea).

MBSE La Ingeniería de Sistemas Basada en Modelos (MBSE) es una metodología formal que utiliza un modelo de sistema integrado como la única fuente de verdad para gestionar la complejidad y el ciclo de vida completo de un producto. Establece una trazabilidad digital directa entre los requisitos, las funciones, la arquitectura y los análisis. Esto permite la Verificación y Validación (VV) continua, integrando el diseño con la simulación para detectar errores de forma temprana y reducir costes. Este enfoque sustituye a los procesos tradicionales basados en documentos, mejorando la coherencia del diseño y la colaboración interdisciplinar..

MDO Multidisciplinary Design Optimization (Optimización de Diseño Multidisciplinar).

MOFLT Mission / Operation / Function / Logic / Technical (Metodología de Airbus DS).

MTOW Maximum Take-Off Weight (Peso Máximo al Despegue).

OAB Operational Architecture Blank (Diagrama de Arquitectura Operacional en Blanco).

OBIGGS On-Board Inert Gas Generation System (Sistema de Generación de Gas Inerte a Bordo).

OC Operational Capability (Capacidad Operacional).

OFV Outflow Valve (Válvula de Salida).

OMG Object Management Group.

PCBD Physical Components Breakdown Diagram (Diagrama de Descomposición de Componentes Físicos).

PERT Program Evaluation & Review Technique.

PRSOV Pressure Regulating and Shut-Off Valve (Válvula de Regulación y Cierre de Presión).

PVMT Property Values Management Tool (Herramienta de Gestión de Valores de Propiedades, una extensión de Capella).

RAMS Reliability, Availability, Maintainability, and Safety (Fiabilidad, Disponibilidad, Mantenibilidad y Seguridad).

ReqIF Requirements Interchange Format (Formato de Intercambio de Requisitos).

RFP Request for Proposal (Solicitud de Propuesta).

SAB System Architecture Blank (Diagrama de Arquitectura de Sistema en Blanco).

SAT Static Air Temperature (Temperatura Estática del Aire).

TAT Total Air Temperature (Temperatura Total del Aire).

TCV Temperature Control Valve (Válvula de Control de Temperatura).

TFG Trabajo de Fin de Grado.

WAI Wing Anti-Ice (Sistema Antihielo de las Alas).

Glosario

Actor Entidad (persona, sistema externo) que interactúa con el sistema o participa en el contexto operacional.

Agregación (Aggregation) Relación "todo-parte" donde las partes pueden existir independientemente del todo.

Análisis comparativo (Trade Study / Trade-off Analysis) Evaluación comparativa de diferentes alternativas de diseño o solución para seleccionar la más adecuada según criterios definidos.

Arquitectura La organización fundamental de un sistema, encarnada en sus componentes, sus relaciones entre sí y con el entorno, y los principios que gobiernan su diseño y evolución.

Arquitectura Física (PA) Descripción de la implementación tecnológica y física del sistema (hardware, software desplegado, conexiones).

Arquitectura Lógica (LA) Descripción del diseño funcional abstracto del sistema, independiente de la tecnología.

Artefacto Producto generado durante el proceso de ingeniería (ej. un modelo, un documento, un diagrama).

Artefacto de Implementación Componente físico que representa una unidad de software real (ejecutable, librería) desplegada en un nodo.

Asignación (Allocation) Relación que indica qué componente es responsable de ejecutar una función.

Bloque (Block) Elemento estructural fundamental en SysML, representa un tipo de componente del sistema, subsistema o entidad externa.

Capacidad Descripción de alto nivel de lo que un sistema (o el entorno operacional) permite hacer a sus actores.

Ciclo de Vida del Sistema Conjunto de fases por las que pasa un sistema desde su concepción hasta su retirada (ej. concepto, desarrollo, producción, operación, mantenimiento, retirada).

Componente Bloque estructural (lógico o físico en Arcadia; o una instancia de un Bloque en SysML) que encapsula funciones y tiene interfaces definidas.

Componente Físico (PC) Elemento de la Arquitectura Física (puede ser un nodo hardware o un artefacto software).

Componente Lógico (LC) Elemento abstracto de la Arquitectura Lógica que agrupa funciones relacionadas.

Composición (Composition) Relación "todo-parte" fuerte donde la existencia de las partes depende del todo.

Conector Elemento que une puertos para permitir intercambios (lógico o físico).

Curación de Modelos (Model Curation) Gestión activa y cuidado de los modelos y entornos de modelado como activos valiosos de una organización.

Despliegue (Deployment) Asignación de artefactos de software (Componentes de Implementación) a nodos de hardware (Nodos Físicos).

Diagrama de Bloques Internos (IBD) Diagrama SysML que muestra las partes internas de un Bloque, sus puertos y los conectores que las unen.

Función Acción, tarea o transformación realizada por el sistema o dentro del contexto operacional.

Generalización Relación "es-un-tipo-de" que establece una jerarquía y permite la herencia.

Gestión del Ciclo de Vida del Producto (PLM) Estrategia y software para gestionar toda la información de un producto a lo largo de su vida.

Herencia (Inheritance) Mecanismo por el cual una subclase adquiere propiedades y comportamientos de su superclase.

Inmaduro (Immature) Estado temprano de desarrollo de un producto, tecnología o proceso, caracterizado por incertidumbre, diseño no finalizado o tecnología no completamente probada.

Intercambio (Exchange) Flujo de información, materia o energía entre funciones (Intercambio Funcional) o a través de conectores entre puertos (Intercambio de Componente).

Interfaz Conjunto de operaciones o flujos definidos que especifican cómo interactuar con un componente a través de un puerto.

Lenguaje de Modelado Conjunto formal de sintaxis y semántica (símbolos, reglas) para representar información sobre un sistema.

Metamodelo Modelo que define la estructura y las reglas de un lenguaje de modelado (define qué elementos y relaciones se pueden usar y cómo).

Metodología Enfoque estructurado o proceso definido (pasos, prácticas, artefactos) para realizar una tarea compleja como la ingeniería de sistemas.

Modelo Representación simplificada de un sistema o proceso, utilizada para entenderlo, analizarlo, comunicarlo o simularlo.

Máquina de Estados (State Machine) Modelo de comportamiento que describe los estados de un sistema y las transiciones entre ellos en respuesta a eventos. Modela el ciclo de vida o comportamiento reactivo.

Nodo Físico (Node) Componente Físico que representa un recurso computacional o de hardware.

Producción (Production - PRO) Fase del ciclo de vida donde el sistema o producto se fabrica o construye para su entrega y despliegue final.

Puerto Punto de interacción definido en el borde de un componente o bloque. (Tipos: Puerto Estándar, Puerto de Flujo).

Punto de Vista (Viewpoint) Definición que especifica cómo generar una Vista (qué mostrar, cómo organizarlo, para quién).

Refinamiento (Refine) Relación de trazabilidad que muestra cómo un elemento (ej. requisito) detalla o deriva de otro.

Requisito Declaración formal de una necesidad, capacidad, característica o restricción que un sistema debe cumplir.

Satisfacción (Satisfy) Relación de trazabilidad que vincula un elemento del modelo con el requisito que cumple.

Simulación Ejecución de un modelo a lo largo del tiempo para observar o analizar su comportamiento dinámico.

Stakeholder (Parte Interesada) Individuo, grupo u organización que tiene interés o se ve afectado por el sistema.

Swimlane Elemento visual en diagramas (ej. Actividad) para agrupar acciones por responsable (actor, rol, componente).

Trazabilidad Capacidad de seguir las relaciones entre diferentes artefactos de ingeniería (ej. de un requisito a una función, a un componente, a un caso de prueba).

Validación Proceso para asegurar que se está construyendo el sistema correcto para las necesidades del usuario y su propósito.

Verificación Proceso para asegurar que el sistema se está construyendo correctamente según sus especificaciones.

Vista (View) Representación específica del modelo del sistema, generada para abordar las preocupaciones de un stakeholder particular.