

**UNIVERSIDAD EUROPEA DE MADRID**

**ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO**

**Máster Universitario en Ingeniería Aeronáutica**

FINAL PROJECT REPORT

**Diseño Sistema ECS mediante metodología MBSE**

**Alfonso Fernández Herrero**

**Curso 2019-2020**

**Título**: Diseño Sistema ECS mediante metodología MBSE

**Autor**: Alfonso Fernández Herrero

**Tutor**: Jesús Martín Sánchez

**Titulación**: Máster Universitario en Ingeniería Aeronáutica

**Curso**: 2020/2021

# Resumen

Mediante el uso de la metodología Model Based System Engineering (MBSE) se realiza un diseño, análisis, validación y verificación de un sistema embarcado ECS. El proyecto abarca desde la edición de requisitos de sistema a partir de una especificación de cliente hasta la generación de herramientas de ayuda a la resolución de incidencias en producción, incluyendo:

* Definición de Requisitos.
* Modelado de sistema.
* Validación.
* Integración del Sistema.
* Análisis de Seguridad operacional.
* Análisis de carga.
* Prueba Funcional de verificación en producción.
* Análisis de cobertura de prueba funcional.
* Método de ayuda a la resolución de incidencias en producción.

**Palabras clave:** MBSE, Sistemas, Requisitos, Automatismo, Modelo.

# Índice de contenidos

[1. Resumen 5](#_Toc40895169)

[2. Índice de contenidos 6](#_Toc40895170)

[3. Figuras 9](#_Toc40895171)

[4. Tablas 10](#_Toc40895172)

[5. Introducción 12](#_Toc40895173)

[5.1. ¿Por qué selecciono el MBSE? 12](#_Toc40895174)

[6. Model Based System Engineering (MBSE) 14](#_Toc40895175)

[6.1. Definición 14](#_Toc40895176)

[6.1.1. “La aplicación de modelos” 14](#_Toc40895177)

[6.1.2. “para respaldar los requisitos” 15](#_Toc40895178)

[6.1.3. “diseño, análisis y verificación y las actividades de validación” 15](#_Toc40895179)

[6.2. Modelo en V 16](#_Toc40895180)

[6.3. El MBSE ofrece 18](#_Toc40895181)

[6.3.1. Flexibilidad ante imprevistos. 18](#_Toc40895182)

[6.3.2. Ingenieros que solo aplican su ingenio. 18](#_Toc40895183)

[6.3.3. Mayor conocimiento de expectativas del cliente. 18](#_Toc40895184)

[6.3.4. Right First Time. 18](#_Toc40895185)

[6.3.5. Tiempos de Desarrollo más cortos. 19](#_Toc40895186)

[6.3.6. Explotación masiva y efectiva de datos 19](#_Toc40895187)

[7. Requisitos de Sistema 20](#_Toc40895188)

[7.1. Requisitos de Alto Nivel (Concepto y Sistema) 20](#_Toc40895189)

[7.2. Requisitos a Nivel de Subsistema 20](#_Toc40895190)

[8. Técnicas de diseño de automatismos 22](#_Toc40895191)

[8.1. Lógica cableada 22](#_Toc40895192)

[8.2. Lógica programada 22](#_Toc40895193)

[8.3. Técnica seleccionada en el Proyecto 23](#_Toc40895194)

[9. Modelo lógico 24](#_Toc40895195)

[9.1. Definición 24](#_Toc40895196)

[9.2. Ejecución 24](#_Toc40895197)

[9.2.1. Modelo de Alimentación 24](#_Toc40895198)

[9.2.2. Modelo de Lógica Cableada 26](#_Toc40895199)

[9.3. Validación 28](#_Toc40895200)

[10. Modelo eléctrico básico 29](#_Toc40895201)

[10.1. Puertas Lógicas en Lógica Cableada 29](#_Toc40895202)

[10.2. Librerías de componentes eléctricos 29](#_Toc40895203)

[10.3. Ejecución 30](#_Toc40895204)

[10.4. Validación 32](#_Toc40895205)

[11. Análisis de seguridad operacional 33](#_Toc40895206)

[11.1. Seguridad Operacional (Safety) 33](#_Toc40895207)

[11.1.1. Amenazas 33](#_Toc40895208)

[11.1.2. Nivel de Seguridad aceptable 33](#_Toc40895209)

[11.2. Analisis de Safety 33](#_Toc40895210)

[11.2.1. Cambios de Diseño en el Analisis de Safety temprano 34](#_Toc40895211)

[11.2.2. Pruebas de no regresión 34](#_Toc40895212)

[11.2.3. Pruebas de Seguridad Operacional 36](#_Toc40895213)

[11.2.4. Resultados 37](#_Toc40895214)

[11.2.5. Acciones Correctoras 38](#_Toc40895215)

[12. Modelo térmico 39](#_Toc40895216)

[12.1. Modelo de generación de calor 39](#_Toc40895217)

[13. Explotación de Datos 41](#_Toc40895218)

[13.1. Monitorización del cableado 41](#_Toc40895219)

[13.2. Prueba de Producción 41](#_Toc40895220)

[13.3. Analisis de Carga 47](#_Toc40895221)

[13.4. Analisis de Cobertura 48](#_Toc40895222)

[13.4.1. Cables redundantes 49](#_Toc40895223)

[14. Troubleshooting 50](#_Toc40895224)

[14.1. Necesidad 50](#_Toc40895225)

[14.2. Solución 50](#_Toc40895226)

[15. Conclusiones 51](#_Toc40895227)

[15.1. Diseño rápido 51](#_Toc40895228)

[15.2. Evita errores de Diseño 51](#_Toc40895229)

[15.3. Analisis de Safety 51](#_Toc40895230)

[15.4. Validación temprana de Integración de Sistemas 51](#_Toc40895231)

[15.5. Diseño acumulativo 52](#_Toc40895232)

[16. Apéndices 53](#_Toc40895233)

[16.1. Anexo: Función para generar la monitorización y control de los cables 53](#_Toc40895234)

[16.2. Anexo: Función para la numeración automática de cableado 54](#_Toc40895235)

[16.3. Anexo: Función para Analisis de carga del Sistema 54](#_Toc40895236)

[16.4. Anexo: Función para comprobar que todo el cableado incluye el bloque Wire 56](#_Toc40895237)

[16.5. Anexo: Función Troubleshooting 57](#_Toc40895238)

[17. Referencias 59](#_Toc40895239)

[17.1. Bibliografía 59](#_Toc40895240)

# Figuras

[Ilustración 1. Tabla de Coste en tiempo y presupuesto utilizados en cada uno de los desarrollos de SpaceX 13](#_Toc40895152)

[Ilustración 2. Modelo en V. 16](#_Toc40895153)

[Ilustración 3. Resultado de simulación de modelo lógico con condición de fuego activada contra condición de fuego desactivada. 25](#_Toc40895154)

[Ilustración 4. Modelo lógico de la lógica cableada. 28](#_Toc40895155)

[Ilustración 5. Diagrama de puertas lógicas en logica cableada. 29](file:///C:\Users\alfon\Documents\DATOS\05.%20TFM\Diseño%20de%20Sistema%20ECS%20mediante%20metodología%20MBSE.docx#_Toc40895156)

[Ilustración 6. Elementos de librería en el modelo electrico 30](file:///C:\Users\alfon\Documents\DATOS\05.%20TFM\Diseño%20de%20Sistema%20ECS%20mediante%20metodología%20MBSE.docx#_Toc40895157)

[Ilustración 7. Arriba: modelo lógico. Abajo: modelo eléctrico. 31](#_Toc40895158)

[Ilustración 8. Resultado de la prueba de Safety en primera iteración. 37](#_Toc40895159)

[Ilustración 9. Modelo eléctrico modificado tras análisis de safety. 38](#_Toc40895160)

[Ilustración 10. Modelo Térmico. 39](#_Toc40895161)

[Ilustración 11. Modelo de Generación de calor. 40](#_Toc40895162)

[Ilustración 12. Resultado de la simulación a 30 minutos. 40](#_Toc40895163)

[Ilustración 13. Modelo de cable. 41](#_Toc40895164)

[Ilustración 14. Modelo de lógica Cableada con bloques de modelo de cable añadido. 41](#_Toc40895165)

[Ilustración 15. Gráficos de ejecución de la prueba de producción. Izquierda: valor de las señales de comando. Derecha: respuesta del sistema y chequeos en verde (satisfactorio) y rojo (fallo). 48](#_Toc40895166)

[Ilustración 16. En amarillo, cables redundantes: W003 y W066. 49](#_Toc40895167)

[Ilustración 17. En amarillo, cables redundantes: W067, W068, W069 y W070 49](#_Toc40895168)

# Tablas

[Tabla 1. Tabla de verdad para el modelo lógico de la alimentación del sistema. 25](#_Toc40895241)

[Tabla 2. Tabla de Verdad con estados más significativos de la lógica cableada. 26](#_Toc40895242)

[Tabla 3. Prueba de producción: pasos y su duración. 44](#_Toc40895243)

[Tabla 4. Código de prueba de producción y momento de inicio de cada paso. 47](#_Toc40895244)

# Introducción

En el presente trabajo se pretende recorrer todas las fases del diseño de un nuevo sistema. Partiendo desde una especificación en la que se listen una serie de necesidades de cliente, estas serán desgranadas en requisitos funcionales que serán la base del diseño.

Una vez claros los requisitos funcionales que deberá cumplir el sistema, se modelará el sistema, empezando por modelos simples que irán aumentando en complejidad. De este modo la complejidad en el modelo alcanzada al final del proyecto asegurará que a la hora de la fabricación el sistema no va a presentar errores de diseño y los posibles errores de funcionamiento se deban a la fabricación.

El diseño mediante modelado nos va a dar la oportunidad de poder desarrollar herramientas de búsqueda de fallos para encontrar y reparar esos errores en la fabricación de una manera rápida y ágil. También veremos la capacidad que tenemos para poder analizar nuestro sistema y optimizarlo o hacerlo robusto al fallo.

## ¿Por qué selecciono el MBSE?

El MBSE es la respuesta desde la Ingeniería de Sistemas al surgimiento de la Cuarta Revolución Industrial. El boom de la tecnología digital de los últimos años hace difícil justificar la no aplicación de la digitalización, simulación y explotación masiva de datos de los diseños aeroespaciales.

La industria aeroespacial tiene una serie de características intrínsecas que lo hacen ideal para la aplicación integral de esta metodología:

* Alto coste del error.
* Necesidad de alta precisión en el diseño.
* Baja cadencia comparada con otras industrias como la automotriz.
* Creciente tendencia al acortamiento de los tiempos de diseño.

Y es que muchas de las actividades definidas en la Ingeniería de Sistemas sin esta tecnología dependen del buen hacer de los ingenieros, que muchas veces no tienen las herramientas para llevarlas a cabo.

Pero los mejores indicadores de que la decisión de selección de esta metodología es la acertada son las decisiones que están tomando los principales fabricantes aeronáuticos:

* **Airbus:** está abriendo centros y proyectos por todo el mundo para la implementación de la metodología, llegando a hacer una implementación relativamente exitosa en el nuevo A350 XWB.
* **Boeing:** al igual que Airbus está apostando con grandes inversiones en obtener personal cualificado y herramientas para la introducción del MBSE en la compañía. Quieren hacer su primera implementación de MBSE con el nuevo Boeing 777X
* **Dassault Systèmes:** ha respondido a esta metodología con el 3DExperience. Catia es la principal herramienta de diseño 3D en la industria aeroespacial. Con 3DExperience pretenden también tener el mismo protagonismo en la disciplina de Sistemas.
* **SpaceX:** el uso de la tecnología de modelado les ha llevado a una rápida, de bajo coste y eficiente implementación de sus cohetes.

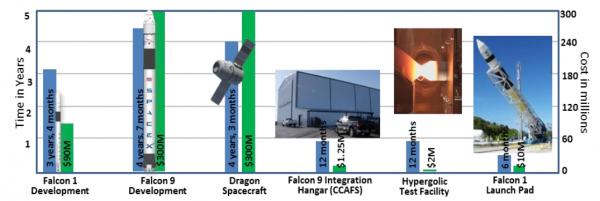


Ilustración . Tabla de Coste en tiempo y presupuesto utilizados en cada uno de los desarrollos de SpaceX

* **SAAB:** ha incluido esta metodología en el diseño de su ultimo caza SAAB Gripen E.

# Model Based System Engineering (MBSE)

La ingeniería no es una ciencia exacta, está basada en estándares y convenciones. Para cada submateria de la ingeniería existe un organismo encargado de regular e indicar cual es la interpretación correcta de los textos técnicos. Cuando hablamos de Ingeniería de Sistemas, es el INCOSE el encargado de realizar estas tareas. Por lo que para conocer que significa la Ingeniería de Sistemas basado en Modelos debemos acudir a él.

## Definición

Según el INCOSE (Organización Internacional de Ingeniería de Sistemas) en 2007, el Model Based System Engineering o en adelante MBSE es:

*“La aplicación de modelos para respaldar los requisitos de sistema, diseño, análisis y verificación y las actividades de validación que comienzan en la fase de diseño conceptual y continúan a lo largo de las fases del ciclo de vida del producto.”*

En lo que sigue iremos desgranado esta definición.

### “La aplicación de modelos”

El termino modelo es muy amplio. No existe una definición clara y única de lo que es un modelo. Si acudimos al INCOSE:

* *“Una representación física, matemática o lógica de un sistema, entidad, fenómeno o proceso.”*
* *“Una representación de uno o más conceptos que pueden realizarse en el mundo físico.”*
* *“Una representación simplificada de un Sistema en algún momento particular en el tiempo o espacio destinado a promover la compresión del Sistema real.”*
* *“Una abstracción de un sistema, dirigida a comprender, comunicar, explicar o diseñar aspectos de interés de ese sistema.”*
* *“Una representación selectiva de algún sistema cuya forma y contenido se eligen en función de un conjunto especifico de inquietudes. El modelo está relacionado con el sistema mediante una asignación explícita o implícita. “*

Todas hacen referencia a un representación o abstracción que permite simplificar un sistema, de manera que sea sencillo de comprender o trabajar. El modelo es la herramienta que usa el ingeniero para obtener un diseño.

Los modelos van desde las fórmulas que explican el comportamiento de un determinado fenómeno físico, a las tablas ingenieriles que en función de los parámetros de entrada dan un resultado de las variables de estado del sistema o al modelo de elementos finitos que permite predecir el comportamiento de una estructura.

### “para respaldar los requisitos”

Los requisitos junto con los modelos son los dos elementos esenciales con los que debe trabajar un Ingeniero. Mientras que los modelos son las herramientas del ingeniero, los requisitos marcan sus objetivos.

Aunque existen muchos tipos de requisitos, en este caso sí que encontramos una definición exacta de que es un requisito:

*“Sentencia que identifica una función, característica o limitación de un producto o proceso, que es univoca, clara, única, consistente, independiente (no agrupada) y verificable, y se considera necesaria para la aceptación del producto o proceso.”*

Si el ingeniero debe de partir de estándares y convenciones, no se explica que toda decisión que tome no esté documentada y justificada. Los requisitos cumplen este papel:

* Cada decisión tomada en el diseño ha de estar justificada con requisitos.
* No puede existir ningún elemento en el diseño que no esté justificado con requisitos.

El ingeniero tiene como objetivo cumplir los requisitos impuestos por el cliente con el diseño. A su vez, ha de generar requisitos que justifiquen cada decisión que tome para cumplir las expectativas del cliente.

### “diseño, análisis y verificación y las actividades de validación”

En este apartado hablaremos de los conceptos de validación y verificación. Ambos corresponden a las actividades que debe de realizar el ingeniero destinadas a asegurar que el cliente recibe el producto que pidió en base a los requisitos impuestos y que el cliente recibe un producto con calidad suficiente.

#### Validación

Cuando hablamos de validación la pregunta que debe de responder el ingeniero es *“¿Estoy fabricando el producto correcto?”*. Esta actividad consiste en comprobar que el diseño que el ingeniero está creando responde a los requisitos impuestos por el cliente.

Si el ingeniero sobredimensiona el diseño, es muy posible que la empresa no saque toda la rentabilidad que pudiera al producto. Si el ingeniero no cumple todos los requisitos impuestos por el cliente, la empresa deberá compensar al cliente.

#### Verificación

Cuando hablamos de verificación la pregunta que debe responder el ingeniero es *“¿Estoy fabricando el producto correctamente?”*. El proceso en el que está englobada esta actividad se le conoce como “Quality Assurance”.

Si hablamos de las etapas de diseño, el ingeniero deberá de comprobar que el producto en diseño una vez fabricado va a trabajar conforme a los requisitos que debe cumplir. Si hablamos de las etapas de producción, el ingeniero ha de comprobar, ya sea en las distintas etapas de producción intermedias o una vez el producto esté terminado, que el producto trabaja conforme a lo diseñado.

## Modelo en V

En este apartado intentaremos responder a “*que comienzan en la fase de diseño conceptual y continúan a lo largo de las fases del ciclo de vida del producto”* de la definición de MBSE del INCOSE.

El modelo en V tiene su origen en el desarrollo software, aunque es válido para toda clase de proyectos de ingeniería de sistemas. Es un proceso que engloba tanto la etapa de diseño como la fabricación e incluso la de mantenimiento, representando la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto.

Se trata de una representación gráfica del ciclo de vida del desarrollo de sistemas en la que se van listando las distintas actividades y entregables que deben producirse durante el desarrollo del producto.

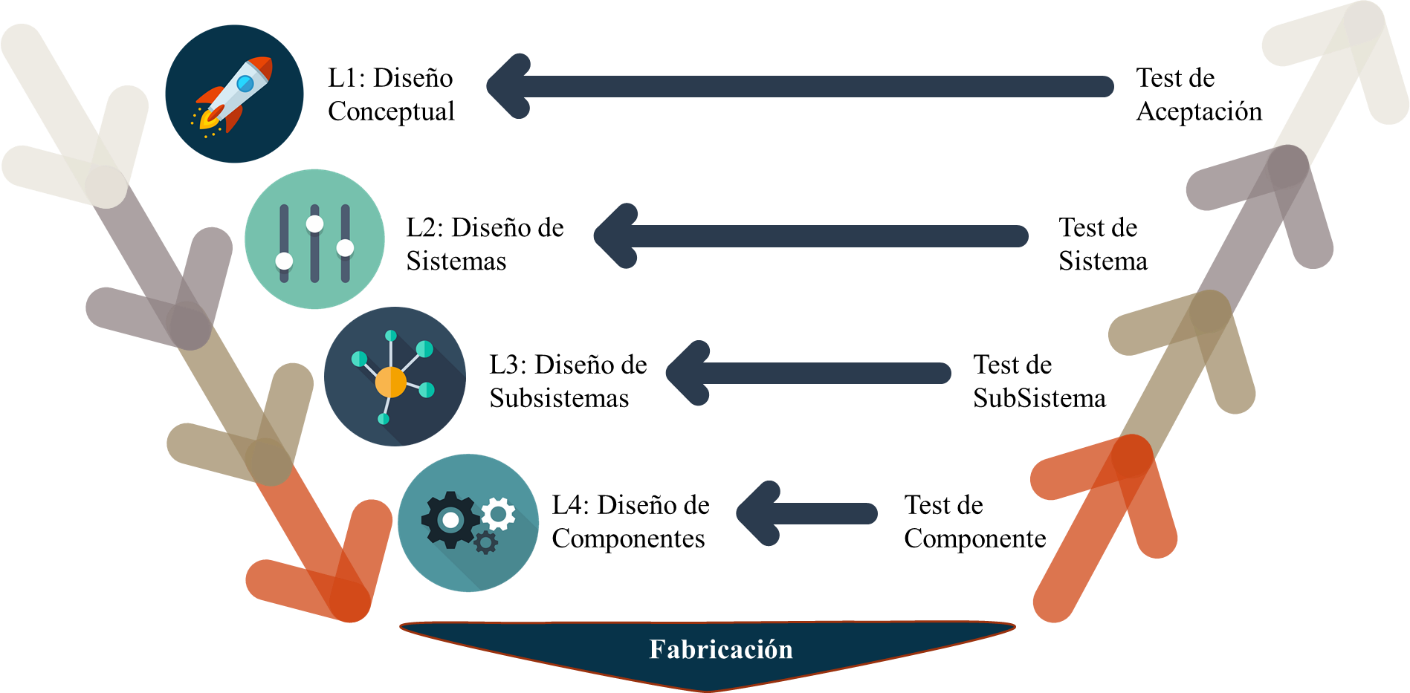


Ilustración . Modelo en V.

En la parte izquierda de la V, la bajada, encontramos todas las actividades relacionadas con el diseño del producto. El vértice de la V representa el proceso de fabricación del producto. En la parte derecha de la V, la subida, encontramos todas las actividades relacionadas con el testeo del producto.

En el modelo de V clásicamente aplicado a la ingeniería de sistemas se definen 5 niveles:

* L0: corresponde al nivel en el que el cliente define los requisitos del producto e impone sus condiciones para aceptarlo.
* L1: corresponde al diseño conceptual del producto y test que se deben de realizar al producto una vez terminado. En este nivel el ingeniero se centra en las interfaces de los sistemas principales.
* L2: corresponde al diseño de los sistemas principales del producto y a los test que se deben de realizar para comprobar que el sistema completo está operativo independientemente a los demás sistemas.
* L3: corresponde al diseño de los subsistemas que componen los sistemas L1 y los test que han de realizarse para comprobar que cada uno opera correctamente. En este nivel el ingeniero se centra las interfaces entre los componentes y equipos que forman los sistemas L2.
* L4: corresponde al diseño de componentes y equipos y los test que se deben de hacer a cada uno.

Así pues, el ingeniero empieza diseñando una “carcasa”, definiendo las condiciones de contorno de su problema y acaba con el diseño del más ínfimo de los componentes.

Sin embargo, cuando se trata de fabricar y testear el sistema el proceso se invierte. El ingeniero empieza comprobando que el componente funciona y acaba comprobando que el producto está listo para la entrega.

Durante todo el recorrido de la V la generación de requisitos, el diseño, la verificación y la validación están presente:

* El proceso diseño o testeo en cada uno de los escalones genera requisitos o habilita la posibilidad del testeo de requisitos para el siguiente.
* En cada escalón hay que realizar una validación, es decir, comprobar que los requisitos para el escalón se han incluido en el diseño y que todo lo diseñado está justificado.
* No es posible avanzar de escalón si no se ha verificado que el diseño es correcto.

Durante el proceso de diseño (bajada), la validación y verificación no garantiza que el producto sea el correcto, pero si evita encontrar muchos problemas durante la etapa de fabricación y testeo. El ingeniero ha de tener muy presente que cuanto más temprano sea el error y más avance el proyecto con él, mayor será el coste para subsanarlo.

Se dice que el diseño de un producto ha sido verificado y validado cuando el primer prototipo ha pasado todos los test que cubren todos los requisitos impuestos al producto. Entonces, ¿cómo podemos hacer una validación y verificación del producto en la etapa de diseño?

La respuesta a esto es el MBSE. Los avances en software de los últimos años han permitido hacer una recreación virtual muy fiel a la realidad de las características, funcionalidades y limitaciones de los productos.

Hoy día el ingeniero tiene herramientas para conocer cómo se va a comportar su producto desde una etapa muy temprana del diseño.

## El MBSE ofrece

### Flexibilidad ante imprevistos.

El modelado (software) de sistemas proporciona una mayor flexibilidad en el diseño. Al existir una continuidad digital, es sencillo y barato comprobar qué posibles implicaciones tienen los cambios en el diseño y adaptar todo el diseño a los cambios en los requisitos.

Con las metodologías analógicas se requiere de un recalculo de todos los parámetros de diseño por cada modificación del diseño y es necesario esperar a la fabricación un prototipo para comprobar si el nuevo diseño se ajusta a las expectativas. Aplicando el modelado, desde un inicio el ingeniero ya tiene capacidad para conocer que transcendencia tiene el cambio desde el primer momento.

### Ingenieros que solo aplican su ingenio.

Desaparecen los documentos descriptivos, ya no es necesario especificar el comportamiento o funciones de un sistema. Toda la información se encuentra en el modelo y el ingeniero tiene que exprimirla. La carga de trabajo dedicada a documentación del diseño se reduce drásticamente y el ingeniero tan solo ha de centrarse en encontrar la mejor solución.

### Mayor conocimiento de expectativas del cliente.

El modelado de las exigencias del cliente ayuda a comprender sus expectativas. El problema del lenguaje escrito es que en muchas ocasiones es abstracto y subjetivo. El modelado usa un lenguaje matemático estándar que no da lugar a interpretaciones.

### Right First Time.

El haber podido simular hasta el más mínimo detalle desde las etapas tempranas del proyecto ayuda a encontrar rápidamente errores mucho antes de fabricar el primer prototipo. Es por esto que se consigue fabricar sin errores de diseño y a la primera.

### Tiempos de Desarrollo más cortos.

La asistencia que prestan los ordenadores permite con las herramientas y conocimientos adecuados, reducir enormemente los tiempos de desarrollo. Es mucho más rápido y preciso calcular las características de una estructura con un software FEM, el cálculo de la aerodinámica de un ala con un CFD o el comportamiento de un determinado circuito eléctrico con PSpice o Matlab/Simulink.

Además, tener a los ingenieros centrados en las soluciones y no sometidos a enormes cargas de trabajo en la generación de documentación ayuda a obtenerlas mucho más rápido y mejores.

### Explotación masiva y efectiva de datos

Uno de los grandes problemas en los procesos de diseño tradicionales el manejo de los datos:

* Mismos datos con distintos valores definidos en distintos procesos del proyecto que generan caos aguas abajo.
* Datos que se pierden a lo largo del proceso de diseño.
* Decisiones de diseño no trazadas ni documentadas.

El proceso de modelado implica a su vez un proceso digital. Este proceso digital garantiza la integridad y trazabilidad de los datos del diseño. Cualquier ingeniero en cualquier momento del proyecto tiene la posibilidad de acceder no solo a parámetros del diseño, sino además a las distintas decisiones con su justificación que se han ido tomando en el proceso de diseño.

# Requisitos de Sistema

## Requisitos de Alto Nivel (Concepto y Sistema)

Por la misión específica para la que está diseñada una aeronave, se requiere de una alta capacidad de computación. Esta alta carga de computación no es constante, por lo que habrá momentos puntuales en que los computadores estén funcionando a pleno rendimiento y se genere excesivo calor.

Por motivos de peso y mantenimiento, se requiere que todos los computadores que van a realizar este trabajo se encuentren juntos en una misma bahía. Esto hace que el calor generado por los computadores se concentre rápidamente en la zona y pueda provocar a la larga graves daños en los equipos.

Se requiere un sistema de ventilación de estos computadores que se adapte al calor generado a fin de mantener una temperatura ideal de 25oC. No se pueden permitir grandes desviaciones de esta temperatura ya que:

* Elevadas temperaturas provocan daños en los computadores.
* Bajas temperaturas provocan bajos rendimientos de los computadores.

En un estudio inicial del calor máximo generado se determina que son necesarios 3 ventiladores. El sistema de control de los tres ventiladores tendrá como única señal de control en cabina un Pulsador Master que además indicará por etiquetado iluminado el estado del sistema.

Además del requisito de la temperatura, este sistema deberá cancelar cualquier flujo de aire en caso de detección de incendio y deberá reaccionar en caso de falla de alguno de los ventiladores iniciando medidas de compensación.

## Requisitos a Nivel de Subsistema

El sistema de control de los ventiladores podrá iniciar los ventiladores de manera independiente.

Las variables de control son:

* La temperatura ambiente en la zona de los computadores:
  + Detección de 26oC.
  + Detección de 28oC.
* La presión negativa producida por el ventilador al absorber aire.
* El estado de encendido o apagado del botón Master en cabina.

La temperatura estará sensorizada por duplicado en la zona de computadores y la presión de cada ventilador estará monitorizada de manera independiente.

La lógica cableada estará alimentada por duplicado y deberá garantizar que al menos un ventilador mantiene el flujo de aire a pesar del fallo de una de ellas.

A la temperatura inferior a 26oC solo habrá encendido un ventilador. En temperaturas entre 26oC y 28oC se deberán de encender dos ventiladores. En temperaturas superiores a 28oC se deberán de encender todos los ventiladores.

En operación a menos de 26oC:

* Cuando se detecte un fallo en el ventilador 1, este deberá de dejar de ser alimentado y se deberá de encender el ventilador 2.
* Si el ventilador 2 también entra en fallo, se deberá iniciar el ventilador 3, dejando de alimentar al ventilador 2.
* En caso de fallo de todos los ventiladores, se deberán de alimentar todos los ventiladores.

En operación entre 26-28oC, cuando se detecte un fallo en el ventilador 1 o 2, se deberá encender el ventilador 3.

En operación superior a los 28oC, todos los ventiladores deberán mantenerse siempre energizados.

En el cockpit, el Pulsador Master tendrá indicación de si existe un fallo de alimentación primaria o alimentación secundaria.

# Técnicas de diseño de automatismos

## Lógica cableada

La Lógica Cableada es una técnica de diseño de automatismos en forma de circuitos cableados, en la que el tratamiento de los datos se efectúa en conjunto con contadores o relés auxiliares que con frecuencia se asocian a temporizadores y contadores.

La lógica cableada incluye funciones de comando y control, de señalización, de protección y de potencia.

Su principal desventaja es que el producto es un automatismo rígido, que realizan tareas de forma secuencial, sin posibilidad de cambiar variables o parámetros. Al contrario que en la lógica programada, su modificación requiere de modificación de mecanismos y recableado, lo que en ocasiones lo hace muy costoso. También existe un coste mayor asociado al hardware necesario para implementarla respecto a la lógica programada.

Pero es un error pensar que la lógica cableada es una técnica obsoleta ya que, en muchas ocasiones, estos sistemas no pueden ser sustituidos por lógica programada. Siempre harán falta elementos de campo, elementos de seguridad y actuadores.

El elemento básico de un automatismo cableado es el contactor para alta potencia y el relé para potencias de control. Ambos se componen básicamente de una bobina, un conjunto magnético y contactos auxiliares normalmente abiertos o normalmente cerrados.

Se emplea en automatismos pequeños, o en lugares críticos, donde la seguridad de personas o maquinas, no puede depender del fallo de una lógica programada.

La normativa sugerida para la implementación de estos automatismos es la IEC1082-1 donde se define la simbología y las referencias identificativas.

## Lógica programada

La Lógica Programada es un tipo de diseño de automatismos que hace uso de circuitos integrados y cuya principal ventaja es la su flexibilidad a la hora de la reconfiguración. Cambiar la configuración de la lógica únicamente requiere de un cambio de software.

La Lógica Programada, en contraposición a la Lógica Cableada, permite fabricar circuitos integrados más o menos genéricos, sin unas funciones lógicas predeterminadas, y definir y almacenar posteriormente su funcionamiento e incluso modificarlo después.

La Lógica Programada está basada en el uso de Dispositivos Lógicos Programables (PLD), que al contrario que las puertas lógicas no tienen una función fija en el momento de su fabricación, si no que su comportamiento depende de la programación que haga el diseñador.

La Lógica Programada, por tanto, requiere de software y de hardware.

## Técnica seleccionada en el Proyecto

Para la ejecución del diseño que nos ocupa utilizaré la lógica cableada. Es la técnica más utilizada para el diseño de Sistemas que no son específicos de Mandos de vuelo, Misión o de Avionica. En nuestro caso se tratándose de un sistema de ECS es la opción que mejor se adapta la aplicación, por coste de diseño y fabricación, por las potencias utilizadas y por la robustez requerida para un sistema crítico de estas características.

# Modelo lógico

## Definición

El diseño lógico de un sistema consiste en desarrollar modelos lógicos que describan la esencia del sistema. Describe de una manera abstracta su funcionamiento con independencia del modo en que se implemente físicamente después.

Al suprimir en el diseño lógico la tecnología, el analista de sistemas tiene mayor libertad para diseñar un modelo que cumpla con los requisitos de sistema.

En este caso utilizaremos un modelo de puestas lógicas para la consecución de los requisitos.

## Ejecución

Partiremos de una tabla de verdad en que aparezcan todos o, al menos, una muestra significativa de los estados de sistema. A partir de esta tabla de verdad mediante algebra booleana obtendremos el modelo de puertas lógicas del sistema

### Modelo de Alimentación

#### Tabla de Verdad

En la tabla de verdad para que el sistema este alimentado por al menos una de las alimentaciones (primaria y/o secundaria) se han definido toda la combinación posible de parámetros de entrada al sistema de alimentación para obtener todos los estados posibles de alimentación.

Estos parámetros de entrada serán:

* Master Pulsado.
* Alimentación Primaria operativa.
* Alimentación Secundaria operativa.
* Detección de fuego.



Tabla . Tabla de verdad para el modelo lógico de la alimentación del sistema.

#### Modelo

A partir de esta tabla de verdad se obtiene el modelo de alimentación conformado por:

* Puerta OR de alimentaciones primaria y secundaria.
* Puerta AND para deshabilitar el Master en caso de fuego.
* Puerta AND que exige que una de las alimentaciones y el Master estén habilitados.

En la siguiente ilustración podemos ver el modelo lógico y el resultado de la simulación con alerta por fuego activa o sin alerta de fuego activada.

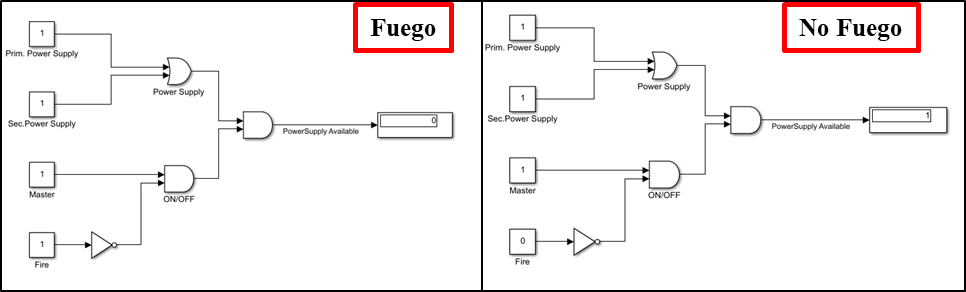


Ilustración . Resultado de simulación de modelo lógico con condición de fuego activada contra condición de fuego desactivada.

### Modelo de Lógica Cableada

#### Tabla de Verdad

Para el modelo de la lógica del sistema se ha creado una tabla de verdad que recoja los estados más significativos y se determina para cada uno de ellos cuál de los ventiladores estará energizado. Las variables de entrada al sistema que vamos a considerar son:

* Alimentación: la alimentación viene determinada el Modelo de Alimentación. En este modelo estará definida como disponible o no.
* Detección de baja presión de aire en cada uno de los ventiladores. Obteniéndose un valor verdadero cuando se detecta baja presión que indica que el ventilador funciona correctamente.
* Fallo en cada ventilador: se considerará fallo cuando un ventilador deba estar operativo y no se detecte presión baja. Este parámetro es el negado de la detección de baja presión.
* Detección de temperaturas ambiente mayores a 26oC y mayores a 28oC.



Tabla . Tabla de Verdad con estados más significativos de la lógica cableada.

Estados significativos identificados:

1. Sistema apagado, pero detección de baja presión en ventiladores y temperaturas superiores a 26 y 28oC: ninguno de los ventiladores ha de estar energizado.
2. Sistema apagado, sin detección de presión baja o alta temperatura: ninguno de los ventiladores energizado.
3. Sistema encendido en su estado de funcionamiento a baja temperatura: tan solo el ventilador 1 está energizado.
4. Sistema encendido en su estado de funcionamiento a baja temperatura, pero con fallo en el ventilador 1: el ventilador 2 deberá estar energizado.
5. Sistema encendido en su estado de funcionamiento a baja temperatura, pero fallo en ventiladores 1 y 2: el ventilador 3 deberá estar energizado.
6. Sistema encendido en su estado de funcionamiento a temperatura superior a 26oC e inferior a 28oC: los ventiladores 1 y 2 deberán estar energizados.
7. Sistema encendido en su estado de funcionamiento a temperatura superior a 26oC e inferior a 28oC: deberá energizarse el ventilador 3 y el ventilador dañado podrá desenergizarse.
8. Sistema encendido en su estado de funcionamiento a temperatura superior a 28oC y con fallo de ventiladores: todos los ventiladores habrán de estar energizados.
9. Sistema encendido en su estado de funcionamiento a temperatura superior a 28oC: todos los ventiladores habrán de estar energizados.

#### Modelo

Necesitaremos una ecuación booleana para cada ventilador. Usando el algebra booleana y siendo:

* Alimentación disponible: A.
* Presión baja en ventilador 1, 2, 3: LP1, LP2, LP3.
* Fallo en ventilador 1, 2, 3: F1, F2, F3.
* Temperaturas superiores a 26 y 28oC: O26, O28.

Obtenemos para cada ventilador las siguientes ecuaciones (subrayado indica negado):

1. Ventilador 1:
   * A\*[F1\*O26\*O28+O26+F1\*F2\*F3]
2. Ventilador 2:
   * A\*[F1\*F2\*O26\*O28+O26+F1\*F2\*F3]
3. Ventilador 3:
   * A\*[F1\*F2\*F3\*O26\*O28+O26\*(F1+F2)+O28+F1\*F2\*F3]

El modelo obtenido es el siguiente:

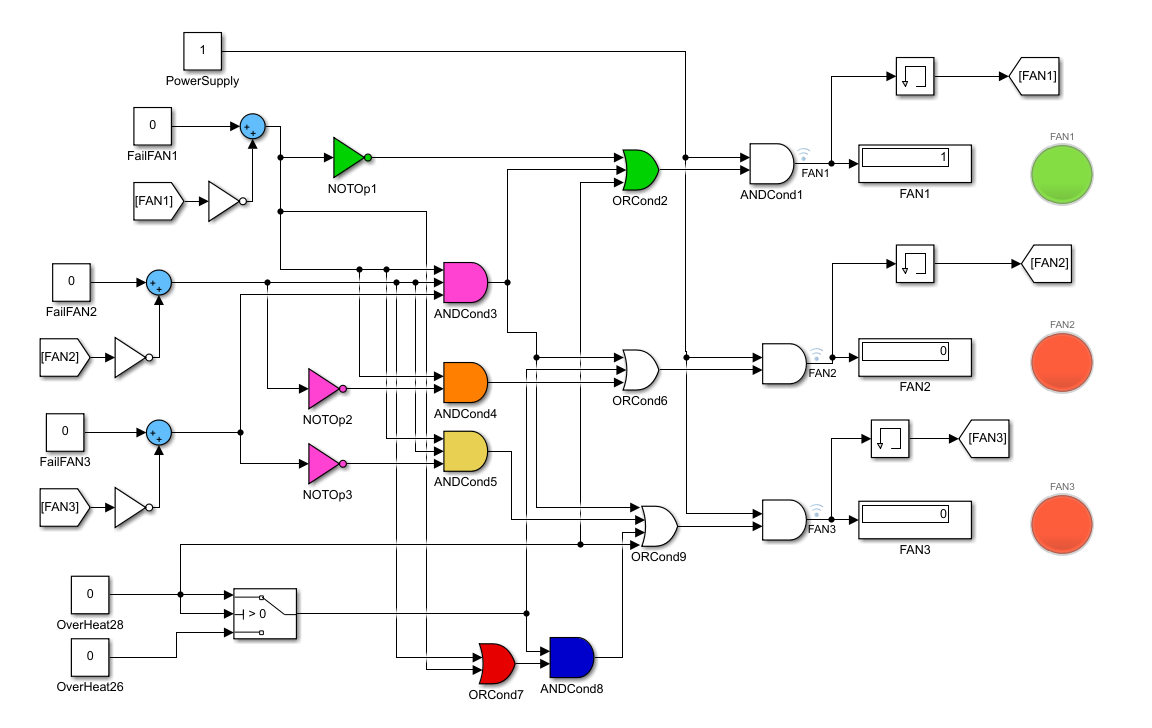


Ilustración . Modelo lógico de la lógica cableada.

El modelo del sistema toma como fallo aquellas situaciones en la que espera que el ventilador funcione, pero sin embargo recibe que el ventilador está parado. Cuando se quiere forzar el fallo basta con forzar siempre un Verdadero por la entrada de monitorización del sistema.

## Validación

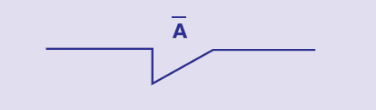
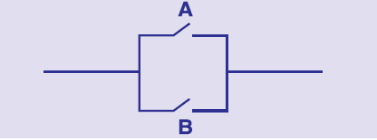
Se comprueba manualmente cada uno de los estados de la tabla de verdad obteniendo los resultados esperados.

# Modelo eléctrico básico

## Puertas Lógicas en Lógica Cableada

Una puerta lógica es la expresión física de la lógica de conmutación. Las puertas lógicas pueden armarse como una red de dispositivos interruptores. Con relés e interruptores podemos conseguir las condiciones de cada puerta lógica:

Ilustración . Diagrama de puertas lógicas en logica cableada.



* Puerta Lógica AND: dos interruptores en serie.
* Puerta Lógica OR: dos interruptores en paralelo.
* Puerta Lógica NOT: un interruptor que se abre cuando esté pulsado.

Con la combinación de estas tres configuraciones de interruptores podremos reproducir con relés e interruptores toda la lógica implementada en el modelo lógico.

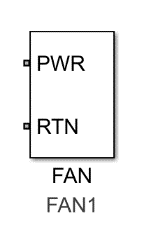
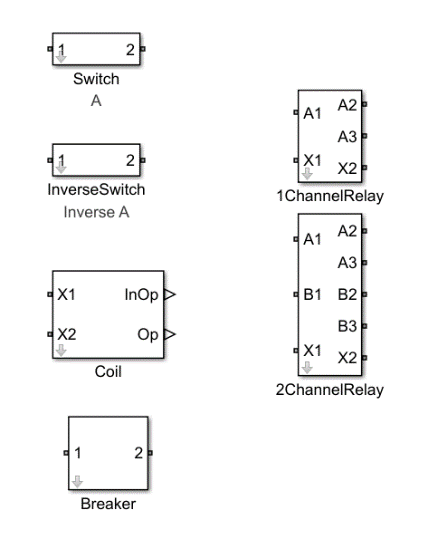
## Librerías de componentes eléctricos

Para garantizar el uso elementos estándar, se ha generado una librería de componentes. El uso de librerías nos garantiza menos errores, mayor mantenibilidad y alta fiabilidad del modelo al construir el mismo sobre los mínimos componentes distintos posibles. Cualquier cambio que hagamos sobre las librerías, tendrá repercusión directa e inmediata sobre todos los elementos del modelo, lo que elimina la posibilidad de error por elementos obsoletos y lo que garantiza que un bug solo haya que resolverlo en un elemento.

Se generan los siguientes componentes:

* Switch: interruptor, se cierra cuando el valor de la señal de control A toma el valor mayor que 1.

Ilustración . Elementos de librería en el modelo electrico



* Inverse Switch: interruptor Switch invertido. Se Cierra cuando el valor de la señal de control A toma el valor 0.
* Coil: determina si la bobina del rele estaría lo suficientemente energizada para conmutar o no los contactos.
* Breaker: Circuit Breaker que se cierra cuando el valor de la señal de control toma el valor mayor que 1.
* XChannelRelay: relé de X canales. Las entradas y salidas de cada canal se indicarán con una letra en el prefijo (excepto X). La bobina está asociada a las entradas X1 de alimentación y X2 de tierra.
  + Cuando la bobina no esté energizada la continuidad en cada canal será de 1 a 2. Ej. *en Canal 1 de A1 a A2*.
  + Cuando la bobina esté energizada la continuidad en cada canal será de 1 a 3. Ej. *en Canal 2 de B1 a B3*.
* FAN: ventilador, modelado como una carga entre los pines PWR y RTN.

## Ejecución

En las siguientes imágenes se muestra la evolución desde el modelo de puertas lógicas al modelo de lógica cableada. Con los colores se representa la equivalencia de elementos de modelo lógico a modelo eléctrico.

Así las puertas lógicas NTOp1 y ORCond2 en verde claro tendrá su equivalente en los relés ORCond2.1 y ORCond2.2 en verde claro también. El botón Master cuenta con indicador de encendido alimentado por la alimentación primaria.

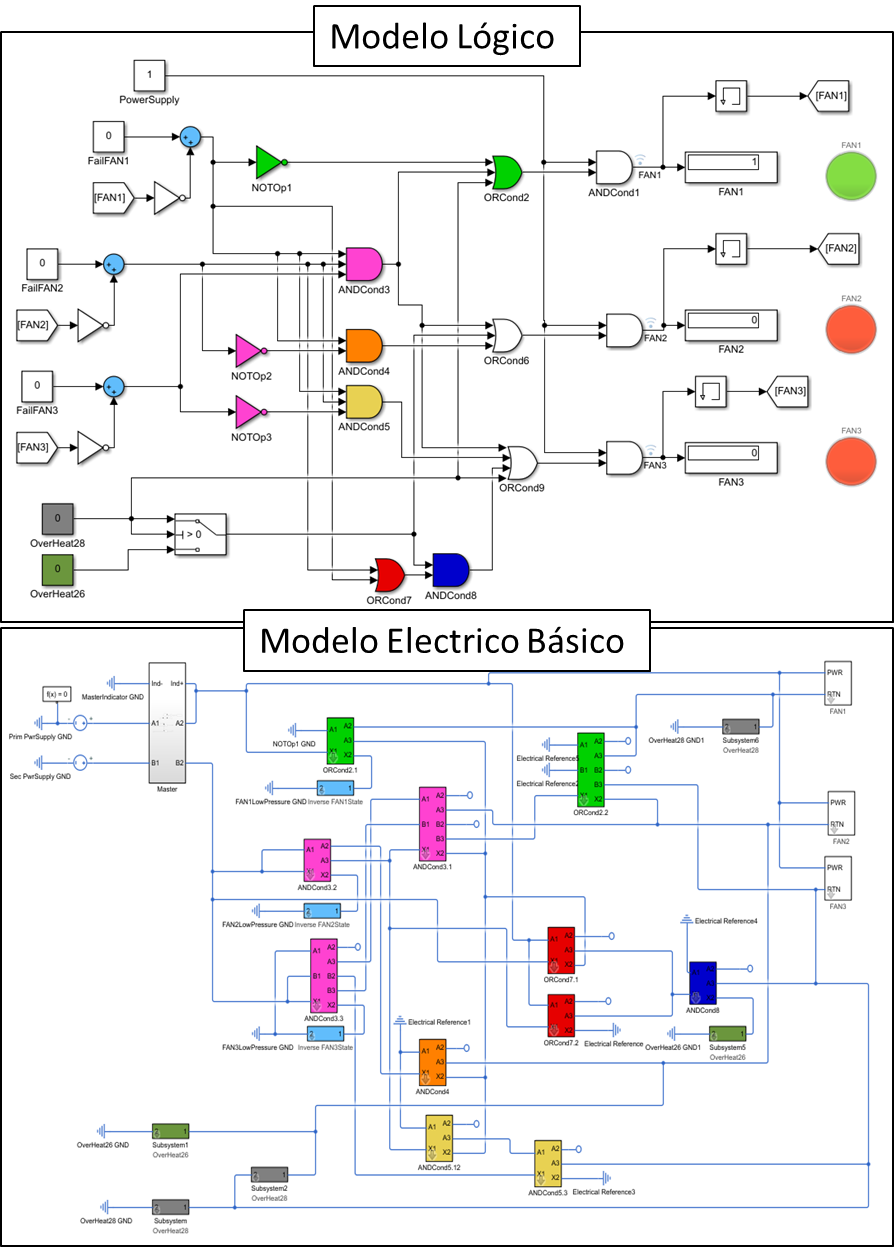


Ilustración . Arriba: modelo lógico. Abajo: modelo eléctrico.

Debido a la inmadurez del diseño, el más mínimo error en el cableado conduce a un comportamiento completamente errático del sistema. Los principales errores que se cometieron durante el diseño fueron:

* Falta de aislamiento de las masas produciendo cortos en el sistema y provocando comportamientos inestables.
* Errores en pineado que alteraban completamente la lógica del sistema.

Es por el primer motivo que se hacen necesarios 3 switch para indicación de temperaturas superiores a los 28oC y dos para la indicación de temperaturas superiores de 26oC.

## Validación

Se comprueba manualmente cada uno de los estados de la tabla de verdad obteniendo los resultados esperados. El resultado obtenido es equivalente al obtenido con el modelo lógico.

Mas adelante con el análisis de Safety se realizarán test de Validación del modelo sobre un estado de diseño más maduro. De esta manera, se puede asegurar que, con los cambios realizados para aumentar la seguridad operacional, no se produce una regresión en la funcionalidad.

# Análisis de seguridad operacional

El sistema diseñado hasta ahora, aunque cumple con todos sus requisitos tiene una alta sensibilidad al fallo de alguno de sus componentes. Durante el proceso de diseño se encuentra que el fallo en el conexionado de alguno de los pines, la generación de lazos de tierra o el fallo de un relé pueden provocar comportamientos inestables que provoquen graves daños.

## Seguridad Operacional (Safety)

Según la OACI se define como seguridad operacional como:

*“Un estado en que el riesgo de lesiones a las personas o daños a los bienes se reduce y mantiene en un nivel aceptable, o por debajo del mismo, por medio de un proceso continuo de identificación de peligros y gestión de medios.”*

La Seguridad Operacional o Safety engloba los procesos y sistemas destinados a reducir el número de accidentes e incidentes derivados en la operación. Aunque aplica a múltiples sectores, tiene una especial relevancia en la industria aeroespacial. Tres pilares fundamentales:

* Definición de niveles de seguridad aceptable.
* Notificación, investigación y análisis de incidencias de seguridad.
* Detección evaluación y mitigación de los riesgos.

La Seguridad Operacional debe ser básicamente proactiva y, más aun, predictiva, con el fin de identificar y prevenir, con la suficiente antelación, y a partir de incidentes menores, las posibles amenazas sobre el sistema para evitarlas o, en caso de ocurrir, mitigar sus consecuencias.

### Amenazas

En este caso se va a hacer un proceder a realizar los cambios oportunos sobre el diseño a fin de mitigar los efectos que tuviera:

* El fallo de cualquiera de los relés que componen el sistema.
* El bloqueo de cualquiera de los relés en su posición de energizado.
* El fallo de alguna de las alimentaciones del sistema.

### Nivel de Seguridad aceptable

El sistema debe de reaccionar a los fallos incluidos en el apartado de amenazas manteniendo siempre alguno de los ventiladores activados y la indicación de estado del sistema.

## Analisis de Safety

### Cambios de Diseño en el Analisis de Safety temprano

Algunas de las debilidades en cuanto a Safety del sistema resultan evidentes previo a la realización de las pruebas:

#### Distribución de Alimentación Primaria y Secundaria

La distribución de alimentaciones de los componentes entre alimentación primaria y secundaria no es segura ya que el fallo de una de ellas puede provocar que no se encienda ninguno de los ventiladores.

El objetivo de los cambios: garantizar que al menos uno de los ventiladores se enciende cuando el sistema pierde alguna de las alimentaciones.

Solución: se realiza una distribución de las alimentaciones de los relés y ventiladores que permite encender los ventiladores 1 y 2 con alimentación primaria y el ventilador 3 con alimentación secundaria.

#### Indicación de estado del sistema

En el modelo preliminar el botón Master, único control en cabina, solo cuenta con indicación de encendido cuando la alimentación primeria está operativa. Esto supone un fallo de Safety grave ya que el piloto pierde indicación de un sistema critico por un fallo.

Para solucionar este problema:

* Se agregan dos relés: PrimFailCond y SecFailCond
* Se agregan dos indicadores entre los pines: [PrimFail+]–[PrimFail-] y [SecFail+]–[SecFail-]

De este modo se añade la indicación de fallo en alimentación primaria, y secundaria y se garantizan indicación del estado de las alimentaciones siempre que el pulsador este activado y exista alguna de las alimentaciones.

### Pruebas de no regresión

Se generan primero cuatro pruebas de no regresión para comprobar que con los cambios que se realizan para aumentar la seguridad operacional se siguen cumpliendo los requisitos funcionales.

1. Prueba del indicador del Master:
   1. Comprobación de indicador ON con pulsador activo y ambas alimentaciones operativas.
   2. Comprobación de indicador de fallo en alimentación primaria con pulsador activo y solo alimentación secundaria operativa.
   3. Comprobación de indicador de fallo en alimentación secundaria con pulsador activo y solo alimentación primaria operativa.
2. Prueba de operación por debajo de los 26oC de temperatura ambiente:
   1. Comprobación de ventilador 1 encendido cuando no existe fallo en el ventilador.
   2. Comprobación de ventilador 2 encendido cuando se fuerza fallo en el ventilador 1.
   3. Comprobación de ventilador 3 encendido cuando se fuerza fallo en ventiladores 1 y 2.
   4. Comprobación de energizado de todos los ventiladores cuando se fuerza fallo en todos los ventiladores.
3. Prueba de operación por encima de los 26oC y debajo de los 28oC de temperatura ambiente:
   1. Comprobación de ventiladores 1 y 2 encendidos cuando no existe fallo en los ventiladores.
   2. Comprobación de encendido de ventiladores 2 y 3 cuando existe un fallo en el ventilador 1.
   3. Reseteo del sistema, eliminando fallo en ventilador 1 y forzado de fallo en el ventilador 2. Comprobación de encendido de ventiladores 1 y 3, y ventilador 2 energizado.
4. Prueba de operación por encima de los 28oC de temperatura ambiente:
   1. Comprobación de encendido de todos los ventiladores cuando no existe fallo en ninguno de los ventiladores.
   2. Comprobación de encendido de ventiladores 2 y 3, y energizado del ventilador 1 cuando el ventilador 1 tiene fallo forzado.
   3. Comprobación de encendido de ventiladores 1 y 3, y energizado del ventilador 2 cuando el ventilador 2 tiene fallo forzado.
   4. Comprobación de encendido de ventiladores 1 y 2, y energizado del ventilador 3 cuando el ventilador 3 tiene fallo forzado.
   5. Comprobación de encendido de todos los ventiladores cuando no existe fallo el sensor de detección de 26oC.

Estas pruebas habrán de ejecutarse con cada cambio de diseño implementado.

### Pruebas de Seguridad Operacional

Las pruebas de Seguridad Operacional estarán destinadas a comprobar si la respuesta del sistema es aceptable ante el fallo/bloqueo de relés y fallos de alimentaciones.

Para ello, se define una prueba genérica que se ejecutará para cada posible fallo o bloqueo de relé o fallo de alimentación. A lo largo de la prueba se forzarán los ventiladores en cada una de las condiciones de temperatura ambiente definidas en los requisitos. Se dará por buena cuando al menos un ventilador se encienda en cada situación de combinación de fallo de ventilador con temperatura ambiente:

1. Comprobación de ventilador encendido sin fallos en ventiladores y temperatura ambiente inferior a los 26oC.
2. Comprobación de ventilador encendido con fallo en ventilador 1 y temperatura ambiente inferior a los 26oC
3. Comprobación de ventilador encendido con fallo en ventilador 2 y temperatura ambiente inferior a los 26oC
4. Comprobación de ventilador encendido con fallo en ventilador 3 y temperatura ambiente inferior a los 26oC
5. Comprobación de ventilador encendido sin fallos en ventiladores y temperatura ambiente inferior a los 28oC y superior a los 26oC.
6. Comprobación de ventilador encendido con fallo en ventilador 1 y temperatura ambiente inferior a los 28oC y superior a los 26oC.
7. Comprobación de ventilador encendido con fallo en ventilador 2 y temperatura ambiente inferior a los 28oC y superior a los 26oC.
8. Comprobación de ventilador encendido con fallo en ventilador 3 y temperatura ambiente inferior a los 28oC y superior a los 26oC.
9. Comprobación de ventilador encendido sin fallos en ventiladores y temperatura ambiente superior a los 28oC.
10. Comprobación de ventilador encendido con fallo en ventilador 1 y temperatura ambiente superior a los 28oC.
11. Comprobación de ventilador encendido con fallo en ventilador 2 y temperatura ambiente superior a los 28oC.
12. Comprobación de ventilador encendido con fallo en ventilador 3 y temperatura ambiente superior a los 28oC.

Se considera un fallo crítico (que puede provocar daños importantes) aquel que provoque que a temperatura superiores a los 26oC no se encienda ningún ventilador. A bajas temperaturas siempre cabe la posibilidad de que al no haber ventilador la temperatura suba y las señales de detección de temperatura fuercen el encendido de algún ventilador. Una vez las señales de detección de temperatura elevada se han activado y los ventiladores siguen sin encenderse, no existe en el sistema posibilidad de reacción.

### Resultados

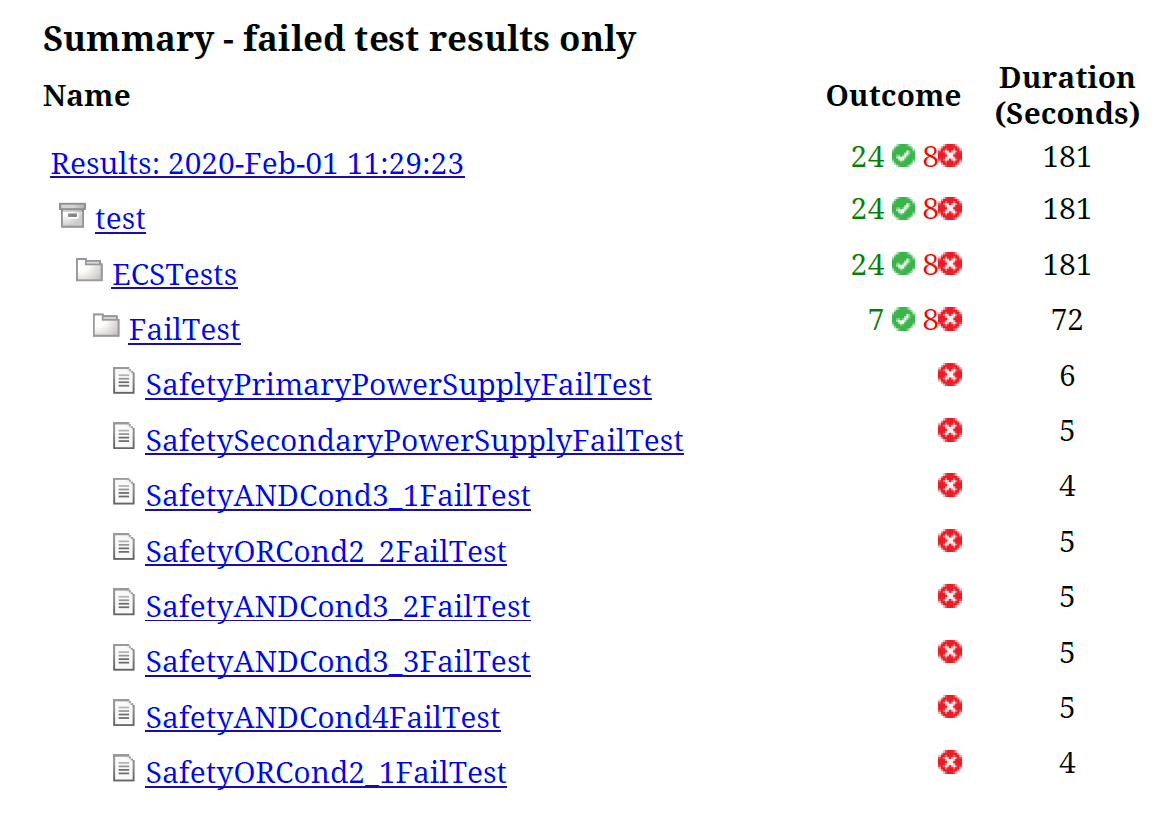


Ilustración . Resultado de la prueba de Safety en primera iteración.

Los resultados de la batería de pruebas arrojan un éxito en 24 de los 32 ejecutados. En las 8 pruebas falladas existe solo en una de ellas un fallo definido como crítico.

Cuando se pierde la alimentación primaria también se pierde la alimentación de los ventiladores 1 y 2 por lo que, si se produce un fallo de este ventilador, el sistema no tiene capacidad de reacción.

### Acciones Correctoras

Para solucionar este problema de seguridad operacional se agrega un relé más que cambia la alimentación del ventilador 2 de la primaria a la secundaria en función de si la alimentación primaria esta activada. Cuando la alimentación primaria falla, la alimentación del ventilador 2 pasa a ser la alimentación secundaria. Sombreado en amarillo en la imagen.

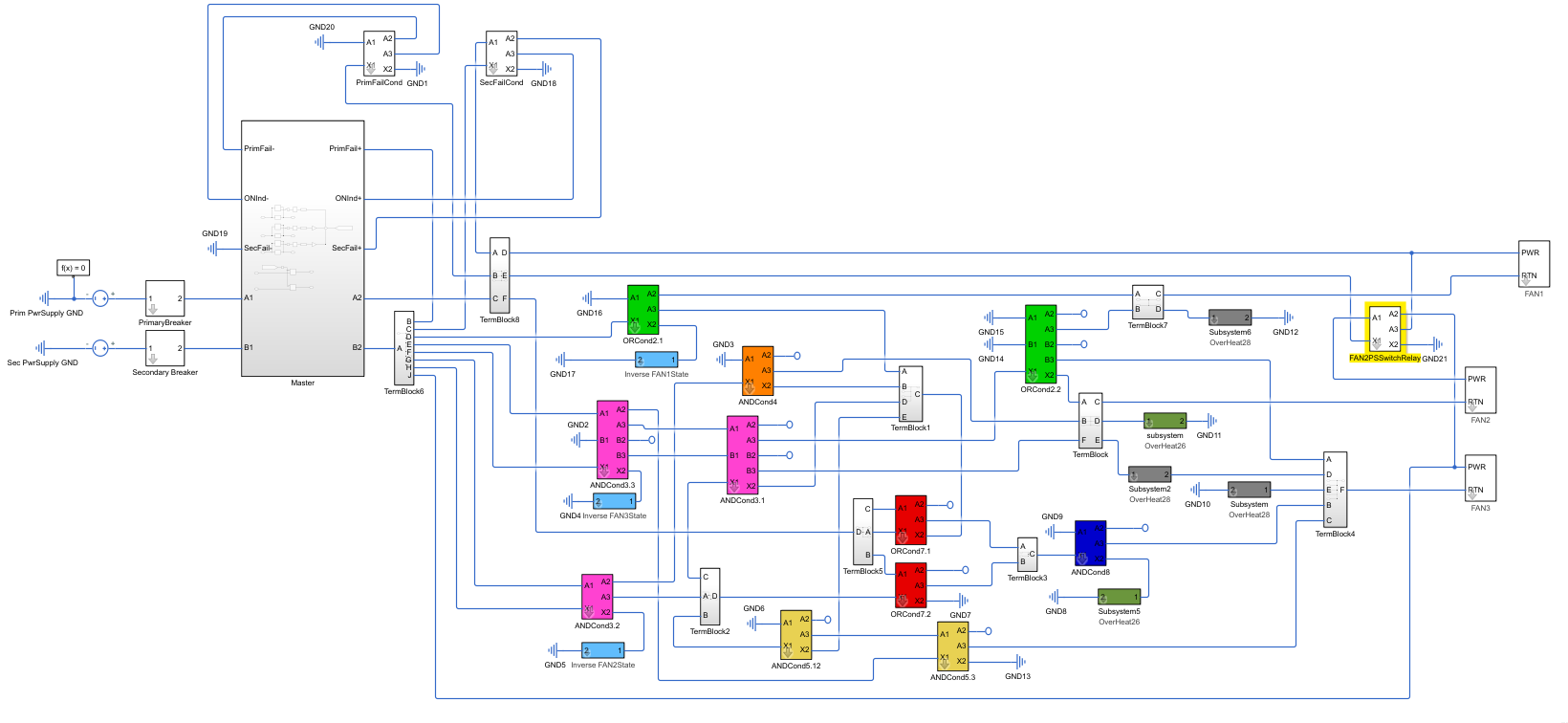


Ilustración . Modelo eléctrico modificado tras análisis de safety.

En este modelo también se han agregado regletas de conexionado (TermBlocks) y los Circuit Breakers de las alimentaciones primarias y secundarias.

# Modelo térmico

## Modelo de generación de calor

Para validar la capacidad del sistema para controlar los ventiladores de manera que la temperatura media quede en torno a los 25oC, se genera un modelo térmico en el que se pueda ver la evolución de la temperatura ambiente en función de la disipación de calor de los ventiladores y la generación de calor de los computadores.

Primero debemos definir el modelo matemático del calor en la habitación:

El calor inicial en la habitación será: Q=Cp\*VAire\*ρ\*T0

La temperatura que vamos a encontrar en la habitación en cada momento: Ti=Q/Cp\*VAire\*ρ

El volumen control del aire donde se encuentran los computadores será de unos 40 m3.

El calor generado por los computadores oscilará entre los 13W y los 29W siendo el consumo nominal de los mismos de 26W. El calor generado tenderá a aproximarse al límite superior, de modo que se ponga a prueba al sistema en la condición de máximo rendimiento.

La capacidad de desalojo del calor de los ventiladores:

* Un ventilador encendido: 18W.
* Dos ventiladores encendidos: 23W.
* Tres ventiladores encendidos: 26W.

El modelo resultante:

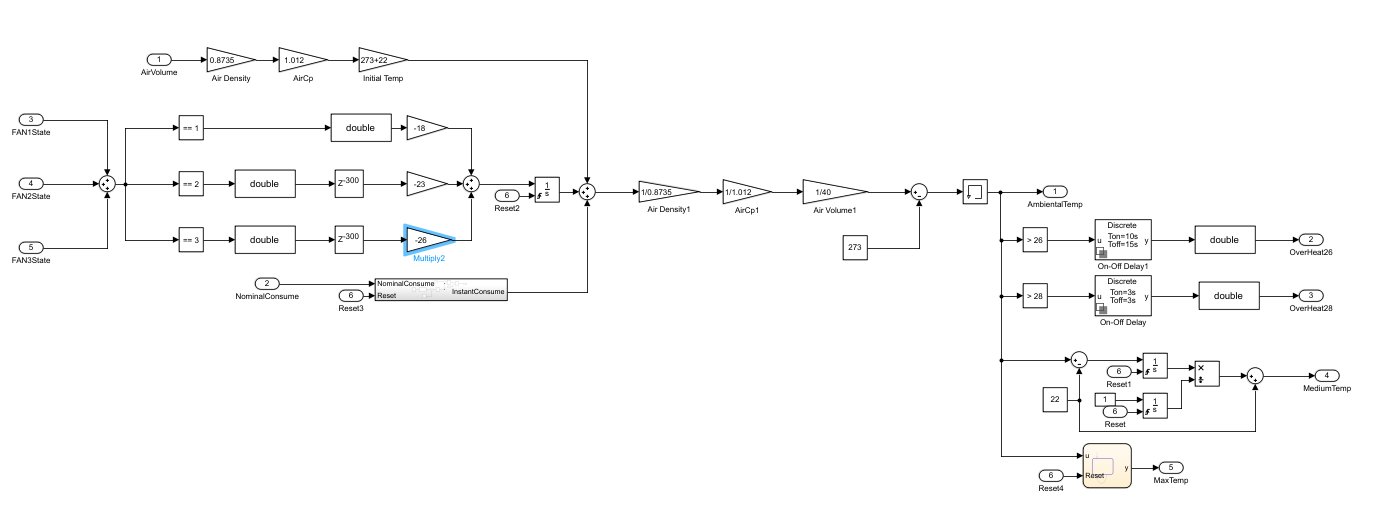


Ilustración . Modelo Térmico.

Se estima que:

* Los ventiladores tardan 0.3 segundo es generar disipación de aire.
* El sensor para la detección de 26oC tarda 10 segundos en la detección y 20 segundo en enfriarse.
* El sensor para la detección de 28oC tarda 3 segundos en la detección y 6 segundo en enfriarse.

La generación de calor de los computadores se ha modelado de manera que no sea constante, siendo aleatoria dentro de los limites nombrados.

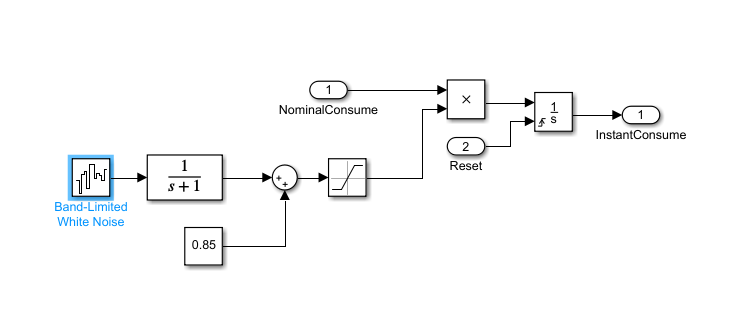


Ilustración . Modelo de Generación de calor.

Después de la una simulación de 30 minutos los datos obtenidos son:

* Temperatura media de 25.9 oC.
* Pico de temperatura de 28.43 oC.

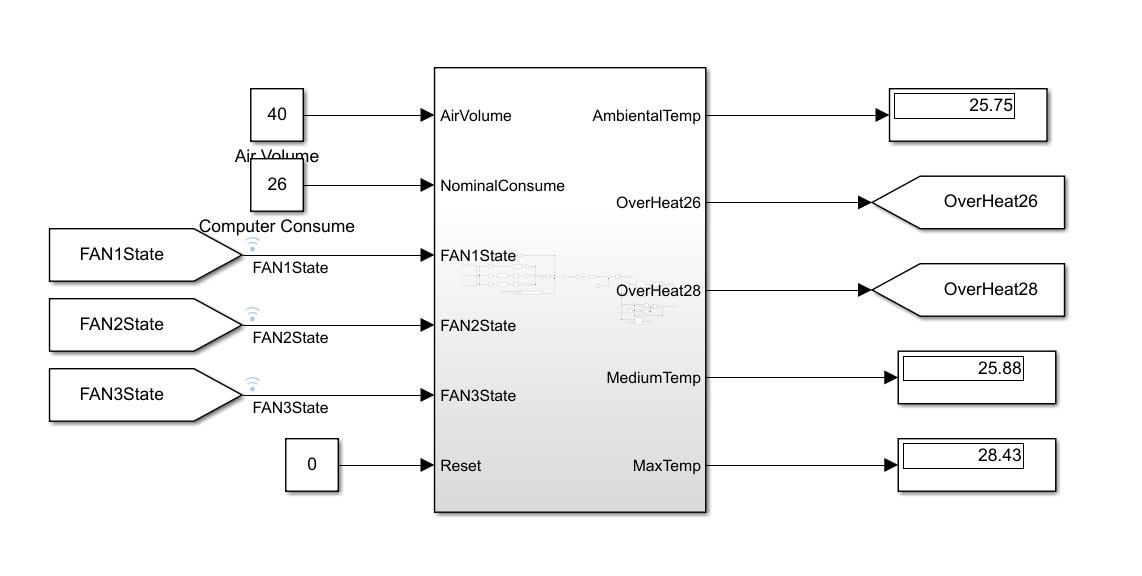


Ilustración . Resultado de la simulación a 30 minutos.

Siendo el calor generado en el modelo superior al que se generaría en la realidad, podemos decir que el sistema completo, control, actuadores e instalación cumplen con sus requerimientos funcionales.

# Explotación de Datos

En los apartados anteriores se ha conseguido un diseño de sistema que cumpla con los requisitos funcionales y de seguridad operacional impuestos. El diseño ya está maduro y puede ser integrado con el resto de los sistemas de avión. Tenemos un modelo de sistema muy completo y del que se puede sacar mucha información.

En este apartado se procederá a la explotación de datos del sistema a fin de sacar información relevante para su optimización del sistema o la generación de herramientas que posteriormente vayan a ser usadas en la producción en serie del avión.

## Monitorización del cableado

Para poder sacar información del modelo es importante conocer cuanta corriente pasa por cada cable en cada fase de actuación del sistema, incluir el efecto del cableado sobre la carga total del sistema y explorar que consecuencia tiene la rotura de un cable.

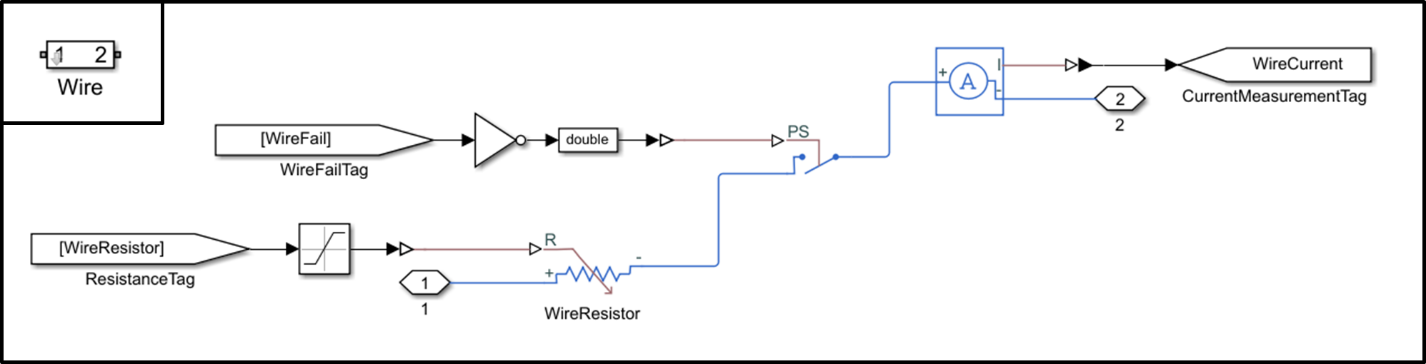
Para esto se ha desarrollado un bloque que añadirá al modelo el efecto del cableado: 

Ilustración . Modelo de cable.

Con este modelo podremos incluir la resistencia que da cada cable, simular la rotura del mismo y monitorizar la corrente que pasa por el mismo.

A cada tramo se le incluye un bloque del tipo “Wire”:

Ilustración . Modelo de lógica Cableada con bloques de modelo de cable añadido.

## Prueba de Producción

Para cada avión producido en serie es necesario comprobar que la totalidad de los elementos funcionales instalados tiene un correcto funcionamiento acorde a los requisitos funcionales. Esto implica excitar todo equipo, componente y cableado instalado en un avión. La manera más clara de hacer esto es editar una prueba que recoja todos los requisitos funcionales y que haga que por cada cable del sistema pase una corriente eléctrica.

La Prueba Funcional de Producción generada para el sistema:

1. Comprobación de indicación de fallo en alimentación secundaria forzando fallo en la alimentación secundaria y con temperatura ambiente por debajo de los 26oC. El ventilador 1 deberá estar operativo.
2. Comprobación de indicación de fallo en alimentación primaria forzando fallo en la alimentación primaria y con temperatura ambiente por debajo de los 26oC. El ventilador 2 deberá estar operativo.
3. Comprobación de operación del ventilador 1 cuando no existe fallo en alimentaciones y la temperatura está por debajo de los 26oC.
4. Comprobación de operación del ventilador 2 cuando hay fallo forzado del ventilador 1 y la temperatura está por debajo de los 26oC.
5. Comprobación de operación del ventilador 3 cuando hay fallo forzado de los ventiladores 1 y 2 y la temperatura está por debajo de los 26oC.
6. Comprobación de que todos los ventiladores quedan energizados cuando se fuerza fallo en todos los ventiladores y la temperatura está por debajo de los 26oC.
7. Comprobación de que los ventiladores 1 y 2 están operativos cuando la temperatura esta entre los 26-28oC y no existen fallos en los ventiladores.
8. Comprobación de que los ventiladores 2 y 3 están operativos cuando la temperatura esta entre los 26-28oC y se ha forzado fallo en el ventilador 1.
9. Comprobación de que todos los ventiladores están operativos cuando la temperatura es superior a los 28oC y no existe fallo en los ventiladores.
10. Comprobación de que los ventiladores 1 y 3 están operativos cuando hay fallo forzado en el ventilador 2 y la temperatura se encuentra entre los 16-28oC.

Estos pasos también corresponden con los requisitos impuestos en el inicio del proyecto, siendo cada uno de ellos independientes. Podemos identificar por tanto donde se cumple cada requisito.

En la siguiente tabla podemos encontrar la prueba en su ejecución paso a paso y la duración esperada de cada uno:

| **Step** | **Title** | **Description** | **Duration** |
| --- | --- | --- | --- |
| 1 | Initial Conditions | Set Up of initial conditions | 0,1 |
| 2 | Primary Power Supply On | Switch On the Primary Power Supply and Master Control | 0,1 |
| 3 | With Secondary Power Supply fail and temperature below 24 degrees, only FAN1 shall be operative | When Secondary Power Supply fail and temperature is below 24 degrees, the FAN1 shall work | 0,1 |
| 4 | Secondary Power Supply On | Switch On the Secondary Power Supply and Master Control | 0,1 |
| 5 | With Primery Power Supply fail and temperature below 24 degrees, only FAN2 shall be operative | When Primary Power Supply fail and temperature is below 24 degrees, the FAN2 shall work | 0,1 |
| 6 | Full Power On | Switch On the Power Supply and Master Control | 0,1 |
| 7 | With no failures and temperature below 24 degrees, only FAN1 shall be operative | When there are no failures and temperature is below 24 degrees, the FAN1 shall work | 0,1 |
| 8 | Force FAN1 failure when temperature is below 24 degress | The failure of FAN1 is forced in this step to check system reaction when temperature is below 24 degress. | 0,1 |
| 9 | With FAN1 Failure and temperature below 24 degrees, only FAN2 shall be operative | When there is a failure of FAN1 and temperature is below 24 degrees, the FAN2 shall work. | 0,1 |
| 10 | Force FAN1 and FAN2 failure when temperature is below 24 degress | The failures of FAN1 and FAN2 are forced in this step to check system reaction when temperature is below 24 degress | 0,1 |
| 11 | With FAN1 and FAN2 Failure and temperature below 24 Degrees, only FAN3 shall be operative | When there is a failure of FAN1 and FAN2 and temperature is below 24 degrees, the FAN3 shall work. | 0,1 |
| 12 | Force FAN1, FAN2 and FAN3 failure when temperature is below 24 degress | The failures of FAN1, FAN3 and FAN3 are forced in this step to check system reaction when temperature is below 24 degress | 0,1 |
| 13 | With FAN1, FAN2 and FAN2 Failure and temperature below 24 Degrees, all FANs shall be energized. | When there is a failure of FAN1, FAN3 and FAN2 and temperature is below 24 degrees, all FANs shall be energized. | 0,1 |
| 14 | Force 26 degrees detection | 26 degrees detection shall be forced. | 0,1 |
| 15 | With no failures and temperature between 26 and 28 degrees, FAN1 and FAN2 shall be operative | When there are no failures and temperature is between 26 and 28 degrees, the FAN1 and FAN2 shall work. | 0,1 |
| 16 | Force FAN1 failure when temperature is between 26 and 28 degrees. | The failure of FAN1 is forced in this step to check system reaction when temperature is between 26 and 28 degrees. | 0,1 |
| 17 | With FAN1 Failure and temperature between 26 and 28 degrees, FAN2 and FAN3 shall be operative | When there is a failure of FAN1 and temperature is between 26 and 28 degrees, the FAN2 and FAN3 shall work. | 0,1 |
| 18 | Force 28 degrees detection | 28 degrees detection shall be forced. | 0,1 |
| 19 | With no failures and temperature above 28 degrees, all FANs shall be operative. | When there are no failures and temperature is above 28 degrees, all FANs shall be operative. | 0,1 |
| 20 | Force FAN2 failure when temperature is between 26 and 28 degrees. | The failure of FAN2 is forced in this step to check system reaction when temperature is between 26 and 28 degrees. | 0,1 |
| 21 | With FAN2 Failure and temperature between 26 and 28 degrees, FAN1 and FAN3 shall be operative and FAN2 shall be energized | When there is a failure of FAN1 and temperature is between 26 and 28 degrees, the FAN1 and FAN3 shall work and FAN2 shall be energized. | 0,1 |

Tabla . Prueba de producción: pasos y su duración.

En la siguiente tabla podemos encontrar el código de establecimiento de variables y chequeo en cada paso de la prueba de producción con el tiempo de inicio del paso:

| **Step** | **Code** | **Step**  **Start Time** |
| --- | --- | --- |
| 1 | Run %% Initialize data outputs. SecPowerSupply = 0; PrimPowerSupply = 0; MasterControl = 0; FAN1Fail = 0; FAN2Fail = 0; FAN3Fail = 0; OverHeat26 = 0; OverHeat28 = 0; | 0 |
| 2 | OnlySecondaryPowerSupplyOn PrimPowerSupply = 0; SecPowerSupply =1; MasterControl = 1; OverHeat26 = 0; OverHeat28 = 0; FAN1Fail = 0; FAN2Fail = 0; FAN3Fail = 0; | 0,1 |
| 3 | CheckFAN1OperationOnlyPrimaryPowerSupply verify(FAN1Ener==0&&FAN2State==1&&FAN3Ener==0&&MasterState==2) | 0,2 |
| 4 | OnlyPrimaryPowerSupplyOn PrimPowerSupply = 1; SecPowerSupply = 0; MasterControl = 1; OverHeat26 = 0; OverHeat28 = 0; FAN1Fail = 0; FAN2Fail = 0; FAN3Fail = 0; | 0,3 |
| 5 | CheckFAN2OperationOnlySecondaryPowerSupply verify(FAN1State==1&&FAN2Ener==0&&FAN3Ener==0&&MasterState==3) | 0,4 |
| 6 | PowerOn PrimPowerSupply = 1; SecPowerSupply = 1; MasterControl = 1; OverHeat26 = 0; OverHeat28 = 0; FAN1Fail = 0; FAN2Fail = 0; FAN3Fail = 0; | 0,5 |
| 7 | CheckFAN1Operation verify(FAN1State==1&&FAN2Ener==0&&FAN3Ener==0&&MasterState==1) | 0,6 |
| 8 | FAN1ForcedFail PrimPowerSupply = 1; SecPowerSupply = 1; MasterControl = 1; OverHeat26 = 0; OverHeat28 = 0; FAN1Fail = 1; FAN2Fail = 0; FAN3Fail = 0; | 0,7 |
| 9 | CheckFAN2OperationWhenFAN1Fail verify(FAN1Ener==0&&FAN2State==1&&FAN3Ener==0&MasterState==1) | 0,8 |
| 10 | FAN2ForcedFail PrimPowerSupply = 1; SecPowerSupply = 1; MasterControl = 1; OverHeat26 = 0; OverHeat28 = 0; FAN1Fail = 1; FAN2Fail = 1; FAN3Fail = 0; | 0,9 |
| 11 | CheckFAN3OperationWhenFAN1and2Fail verify(FAN1Ener==0&&FAN2Ener==0&&FAN3State==1&&MasterState==1) | 1 |
| 12 | FAN3ForcedFail PrimPowerSupply = 1; SecPowerSupply = 1; MasterControl = 1; OverHeat26 = 0; OverHeat28 = 0; FAN1Fail = 1; FAN2Fail = 1; FAN3Fail = 1; | 1,1 |
| 13 | CheckFANOperationFAN3Fail verify(FAN1Ener==1&&FAN2Ener==1&&FAN3Ener==1&&MasterState==1) | 1,2 |
| 14 | OverHeat26Degrees PrimPowerSupply = 1; SecPowerSupply = 1; MasterControl = 1; OverHeat26 = 1; OverHeat28 = 0; FAN1Fail = 0; FAN2Fail = 0; FAN3Fail = 0; | 1,3 |
| 15 | CheckOverHeat26DegreesFANOperation verify(FAN1State==1&&FAN2State==1&&FAN3Ener==0&MasterState==1 | 1,4 |
| 16 | OverHeat26DegreesFAN1Fail PrimPowerSupply = 1; SecPowerSupply = 1; MasterControl = 1; OverHeat26 = 1; OverHeat28 = 0; FAN1Fail = 1; FAN2Fail = 0; FAN3Fail = 0; | 1,5 |
| 17 | CheckOverHeat26DegreesFANOperationFAN1Fail verify(FAN1Ener==0&&FAN2State==1&&FAN3State==1&&MasterState==1) | 1,6 |
| 18 | OverHeat28Degrees PrimPowerSupply = 1; SecPowerSupply = 1; MasterControl = 1; OverHeat26 = 1; OverHeat28 = 1; FAN1Fail = 0; FAN2Fail = 0; FAN3Fail = 0; | 1,7 |
| 19 | CheckOverHeat28DegreesFANOperation verify(FAN1State==1&&FAN2State==1&&FAN3State==1&&MasterState==1) | 1,8 |
| 20 | OverHeat26DegreesFAN2Fail PrimPowerSupply = 1; SecPowerSupply = 1; MasterControl = 1; OverHeat26 = 1; OverHeat28 = 0; FAN1Fail = 0; FAN2Fail = 1; FAN3Fail = 0; | 1,9 |
| 21 | CheckOverHeat26DegreesFANOperationFAN2Fail verify(FAN1State==1&&FAN2Ener==1&&FAN3State==1&&MasterState==1) | 2 |

Tabla . Código de prueba de producción y momento de inicio de cada paso.

## Analisis de Carga

Analizando la carga que supone el sistema sobre el sistema eléctrico del avión durante la ejecución de la prueba de producción:

* Referencia del cable de alimentación primaria: W040.
* Referencia del cable de alimentación secundaria: W041.
* La máxima potencia consumida por el sistema: 125W.
* La corriente en el momento de máximo consumo de potencia:
  + W040: 2 amperios
  + W041: 2.5 amperios
* El consumo en el momento máximo de consumo de potencia del sistema:
  + W040: 54.6 Watios.
  + W041: 70W Watios.
* Momento en que se produce el pico de potencia: segundo 1.3.
* Paso de la prueba en que se produce el pico de potencia: forzado de la detección de temperatura entre 26 y 28oC
* Potencia consumida media: 69Watios.
* Valor más frecuente de potencia consumida: 66W:

Analizando cada uno de los cables de alimentación obtenemos que:

* Para la alimentación primaria (W041):
  + La máxima potencia es de: 54.6 W.
  + El máximo amperaje es de: 2 A.
  + Se produce en el momento: segundo 1.7.
* Para la alimentación secundaria (W041):
  + La máxima potencia es de: 77.8 W.
  + El máximo amperaje es de: 2.8 A.
  + Se produce en el momento: segundo 0.1.

## Analisis de Cobertura

El análisis de cobertura consiste en comprobar cuantos elementos del sistema son excitados mediante la prueba funcional propuesta. Por un lado tenemos una prueba funcional que asegura que los elementos funcionales del sistema cumplen con su función y por el otro vamos a analizar la corriente que circula por el cableado que forma la instalación. Para ello, se ejecuta la prueba funcional y se comprueba que la totalidad del cableado ha sido excitado en algún momento de la prueba.

En la siguiente imagen podemos ver los valores que toman parámetros de controla a la entrada del modelo (izquierda) contra la respuesta del sistema por sus parámetros de salida (derecha). En verde cada una de las comprobaciones realizadas en el tiempo de los parámetros de salida y que cumplen con los solicitado en la prueba funcional. El resultado de la prueba fue satisfactorio y la totalidad del cableado fue excitado.

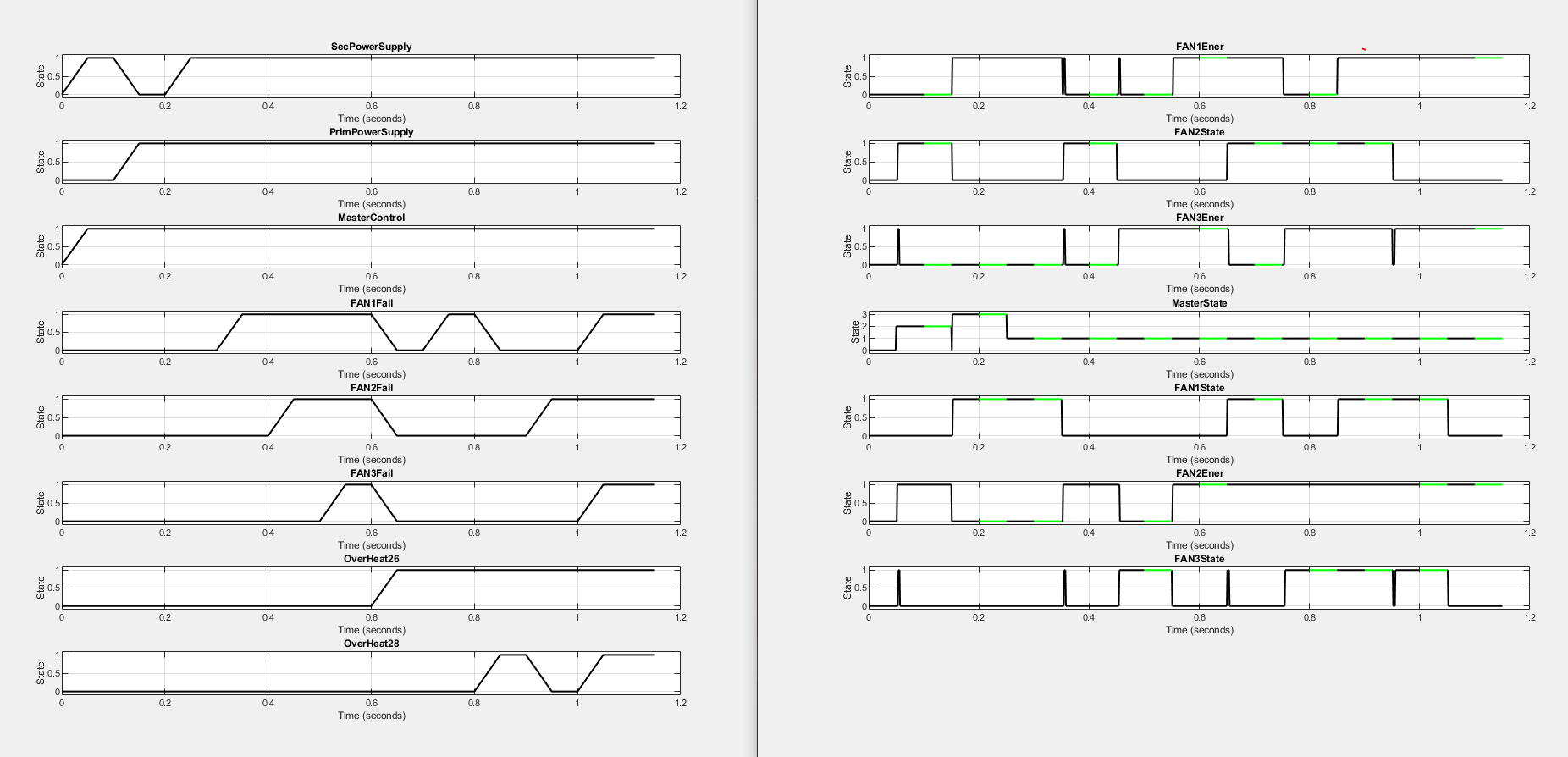


Ilustración . Gráficos de ejecución de la prueba de producción. Izquierda: valor de las señales de comando. Derecha: respuesta del sistema y chequeos en verde (satisfactorio) y rojo (fallo).

También es interesante, para la ayuda a la búsqueda de errores de producción en taller, conocer qué cables afectan a cada paso de la prueba de producción. Esto se puede lograr mediante la ejecución de la prueba de producción forzando que uno de los cables esté dañado cada vez. De este modo obtenemos una base de datos amplia para conocer el comportamiento del sistema ante el fallo de cableado.

### Cables redundantes

De la batería de pruebas ejecutadas con cables rotos se pueden detectar cables que no afectan al resultado. Sin embargo, estos cables sí que son excitados en la ejecución de la prueba sin cable roto. Los cables que no afectan al resultado del Test de Producción por ser redundantes:

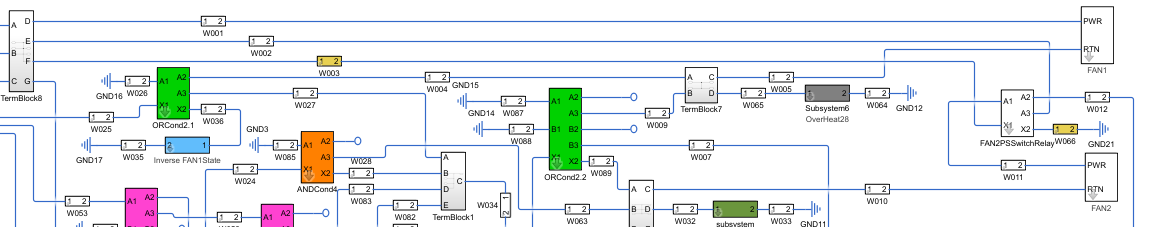
* W003 y W066: asociados al relé de conmutación de la alimentación del FAN2.  
  

Ilustración . En amarillo, cables redundantes: W003 y W066.

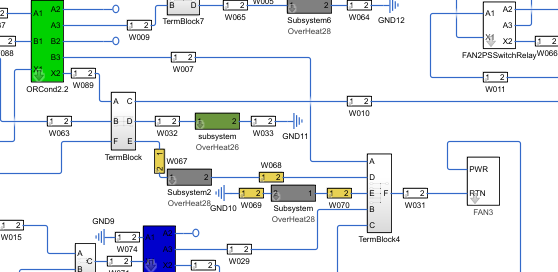
* W067, W068, W069, W070: cables para forzar tierra los TermBlock0 y TermBlock4 en caso de temperaturas superiores a 28oC y asegurar así el encendido de los ventiladores aun con fallos en los elementos de la lógica cableada.  
  

Ilustración . En amarillo, cables redundantes: W067, W068, W069 y W070

# Troubleshooting

## Necesidad

Tras el surgimiento de la metodología “*Lean Manufacturing*” en las líneas de ensamblaje final de las aeronaves se trabaja a contrarreloj. A la vez que se está montando los sistemas, estos se van probando. Ciertas zonas, por el flujo natural del ensamblaje se van volviendo inaccesibles, por eso es importante que estas zonas queden totalmente probadas y poder descartar que los errores están en estas zonas.

También es posible que, al estar realizándose trabajos de ensamblaje y pruebas funcionales simultáneamente, existan conflictos. Por tanto, a los fallos en el montaje se le suman accidentes que provoquen deterioros en los sistemas y pruebas funcionales simultaneas que provoquen estados de los sistemas incompatibles.

El tiempo en encontrar los errores se vuelve algo crucial. El tiempo gastado en la búsqueda del fallo hace aumentar exponencialmente el coste del error:

* No se pueden ejecutar trabajos en la zona de avión afectada por el error.
* Recursos dedicados a las tareas improductivas de búsqueda de información que retrasan otras tareas.

La consecuencia son retrasos en las entregas de avión o sobrecostes para reparar el sistema a tiempo.

Esto hace que cualquier reducción de tiempo en las tareas de búsqueda del error supongan un gran alivio en la fase de ensamblado final de una aeronave.

## Solución

En apartados anteriores, se generó una prueba de producción que excitaba todos los cables del sistema. Esta prueba fue ejecutada para la rotura de cada cable del sistema. Disponemos entonces de una enorme base de datos del comportamiento del sistema ante la rotura de uno de sus cables.

Conociendo los pasos en que la prueba ha fallado y comparando con los resultados obtenidos en las pruebas podemos obtener todos los cables posibles causantes del error y con la conectividad de estos, los elementos susceptibles de estar fallando.

El mismo ejercicio de analizar la cobertura también nos vale para encontrar fallos. Gracias a la digitalización del trabajo, aunque con más esfuerzo evitamos repetir esfuerzos.

# Conclusiones

## Diseño rápido

Descarga al ingeniero de la realización de cálculos y comprobaciones que con otras metodologías se vería obligado a hacer. Además, permite la reutilización de los diseños, adaptando elementos o configuraciones a las necesidades del nuevo producto.

## Evita errores de Diseño

La gran cantidad de comprobaciones y datos que se pueden hacer y obtener de manera sencilla ayudan al ingeniero a conocer cada aspecto de su diseño, tanto implementados a conciencia como aquellos que son involuntarios y consecuencia de la toma de decisiones ingenieriles.

Esta metodología da la posibilidad de ejecutar comprobaciones masivas. Excitando el modelo del sistema de todas las maneras posibles para encontrar tanto los comportamientos predecibles como los que no y a su vez analizarlos para ver si se trata de un error o un comportamiento aceptable del sistema.

## Analisis de Safety

El MBSE hace posible de un vasto análisis de Safety sin haber fabricado el sistema. La capacidad de cálculo del ordenador para simular el comportamiento permite valorar todas las posibilidades de fallo en muy poco tiempo. Tan solo requiere definirlas.

Esta es una actividad que resulta imposible sin el modelado a no ser que se asuma el alto coste en recursos con alta cualificación para el cálculo del comportamiento.

Es una actividad sin el modelado es inasumible para sistemas poco críticos pero importantes.

## Validación temprana de Integración de Sistemas

En este caso práctico se han puesto a trabajar en conjunto dos modelos de materias distintas:

* Modelo de electrónica de control.
* Modelo térmico del ambiente.

De esta forma se ha podido validar que el diseño propuesto para controlar el sistema de disipación de calor es suficiente para controlar la temperatura ambiente a una media de 26oC y picos no mayores a 30oC.

El trabajo en el diseño de un avión está dividido por sistemas y tecnología. Así, muchas veces la persona que trabaja en la estructura del avión no tiene una visión de lo que hace el analista de sistemas o no es posible enfrentar una dimensión del diseño con la otra para medir el grado de integración.

Por el método tradicional es difícil llegar al grado de certeza alcanzado con el método aplicado en este trabajo. Con la simulación en tiempo real de la interacción del sistema de control de ventiladores con la temperatura ambiente, podemos llegar a hacer una validación temprana del sistema, siempre que el modelo térmico y el eléctrico sean suficientemente próximos a la realidad.

## Diseño acumulativo

Como hemos visto en este trabajo, el modelo ha ido aumentando en complejidad en cada una de las fases. A un mismo modelo del sistema se le van acoplando los distintos modelos de cada una de las áreas implicadas. De este modo obtenemos un diseño validado y verificado en virtual con un grado de fidelidad a la realidad muy alto y con una reutilización máxima del trabajo realizado.

# Apéndices

## Anexo: Función para generar la monitorización y control de los cables

%% This script will generate all wire monitorization and control elements

%Get All Wires Path

Model='PressureMonitorModel\_SafetyAnalysisWithWires/WireMonitors&Control';

WireList=split(string(sort(find\_system(bdroot,'FollowLinks','on', ...

'LookUnderMasks','on','ReferenceBlock','ElectricalElements/Wires/Wire'))),"/");

%Get all Wires Block Name

WireList=WireList(:,2);

%Mux for monitorization of wires current

add\_block('simulink/Commonly Used Blocks/Mux',[Model '/MonMux'],'Inputs',num2str(size(WireList,1)), ...

'Position',[70+50 0 70+58 20+30\*(size(WireList,1)-1)])

%Demux to control the wires resistance

add\_block('simulink/Commonly Used Blocks/Demux',[Model '/ResDemux'],'Outputs',num2str(size(WireList,1)), ...

'Position',[(70+50)+900 0 70+58+900 20+30\*(size(WireList,1)-1)])

for i=1:1:size(WireList,1)

%% Monitorization

add\_block('simulink/Signal Routing/From',[Model '/Mon' char(WireList(i))], ...

'GotoTag',[char(WireList(i)) 'Current'], 'position', [0 0+30\*(i-1) 70 20+30\*(i-1)])

add\_line(Model,['Mon' char(WireList(i)) '/1'],['MonMux/' num2str(i)]);

%% Fail

add\_block('simulink/Signal Routing/Goto',[Model '/Con' char(WireList(i))], ...

'GotoTag',[char(WireList(i)) 'Fail'], 'position', [70+58+400+50 0+30\*(i-1) 70+58+400+50+70 20+30\*(i-1)], ...

'TagVisibility','global')

add\_block('simulink/Commonly Used Blocks/Constant',[Model '/' char(WireList(i)) 'Fail'], ...

'Value','0', 'position', [70+50+400 0+30\*(i-1) 70+50+400+30 20+30\*(i-1)])

add\_line(Model,[char(WireList(i)) 'Fail/1'],['Con' char(WireList(i)) '/1']);

%% Resistor

add\_block('simulink/Signal Routing/Goto',[Model '/Res' char(WireList(i))], ...

'GotoTag',[char(WireList(i)) 'Resistor'], 'position', [70+58+900+50 0+30\*(i-1) 70+58+900+50+70 20+30\*(i-1)], ...

'TagVisibility','global')

add\_line(Model,['ResDemux/' num2str(i)],['Res' char(WireList(i)) '/1']);

end

## Anexo: Función para la numeración automática de cableado

function [] = WireNumeration(Block)

% Automatic Wire Numeration generation

% This function should be asociated to a Wire Block callback

T=split(string(get\_param(gcbh,'Parent')),"/"); %Get block path without block name

if ~bdIsLibrary(T(1)) %Not activate when the block is in the library file

BlockName=get\_param(gcbh,'Name');

if length(BlockName)>1

if length(BlockName)~=4 || BlockName(1)~='W' || ~isfinite(str2double(BlockName(2:end)))

% Block Name is not standard

set\_param(Block,'Name',NewName(BlockName)); % Set a new name

end

else

% Block Name is not standard

set\_param(Block,'Name',NewName(BlockName)); % Set a new name

end

end

end

function [NName] =NewName(BlockName)

%% Function to generate a new unique name

%Get all wires

A=split(string(sort(find\_system(bdroot,'FollowLinks','on', 'SearchDepth','1', ...

'LookUnderMasks','on','ReferenceBlock','ElectricalElements/Wires/Wire'))),"/");

NName="";

for i=size(A,1):-1:1

B=char(A(i,2));

% Looking for last named wire

if length(B)==4 && B(1)=='W' && ~isnan(str2double(B(2:end))) && string(B)~=BlockName

% New Wire will be named as the next of current last named wire.

NName=sprintf('W%03d', str2double(B(2:end))+1);

break

end

end

end

## Anexo: Función para Analisis de carga del Sistema

function [PMax, CurrentsInPMax, PowersInPMax, TimeInTMax, MeanP, MedianP, ...

PowInPMaxeWire, CurInPMaxeWire, TInPMaxeWire]=...

ElecLoadAnalysis(WC, PrincipalPSWires, PrincipalWiresV)

% This function analyzes the load of an electrical system where there are

% not AC voltage

%% INPUTS

% WC: each wires currents measured. First row are the timeseries of the

% test without cut wires. [timeseries 2D array].

% PrincipalPSWire: wires of the System Principal Power Supplies

% [string 1D array].

% PrincipalWiresV: System Principal Power Supplies Voltage in DC in Volts

% [double 1D array].

%% OUTPUTS

% PMax: max Total Power Comsumtion in Wats.

% CurrentsInPMax: currents in wires [PrincipalPSWire] at the moment of max power

% comsumtion in Amp.

% PowersInPMax: power in wires [PrincipalPSWire] at the moment of max power

% comsumtion in Wats

% TimeInTMax: moment of max Total Power Comsumtion in seconds.

% MeanP: mean of Total Power Comsumtion

% MediamP: median of Total Power Comsumtion

% PowInPMaxeWire: max power comsumtion for each wire [PrincipalPSWire] in Wats.

% CurInPMaxeWire: max current for each wire [PrincipalPSWire] in Amps.

% TInPMaxeWire: moment in which max power comsumtion for each wire

% [PrincipalPSWire] is got.

%% Takes only the test executed without a cut wire.

NoWireFaultWC=WC(1,:);

%% Which timeseries correspond to the Power Supply Principal Wires

for i=1:length(PrincipalPSWires)

for j=1:numel(NoWireFaultWC)

if NoWireFaultWC(j).Name==PrincipalPSWires(i)+"Current"

PSW(:,i)=NoWireFaultWC(j).Data;

PSWT(:,i)=NoWireFaultWC(j).Time;

break

end

end

end

%% Total power consumed at all times

PSum=PSW(:,1)\*PrincipalWiresV(1);

for i=2:size(PSW,2)

PSum=PSum+PSW(:,i)\*PrincipalWiresV(i);

end

%% Max Power consumtion

[PMax, MaxIdx]=max(PSum);

MeanP=mean(PSum);

MedianP=median(PSum);

TimeInTMax=PSWT(MaxIdx,1);

%% Power consumed by each wire of the System Principal Power Supplies at

% the moment of Max Power comsumtion

CurrentsInPMax=zeros(size(PSW,2),1);

PowersInPMax=zeros(size(PSW,2),1);

for i=1:size(PSW,2)

CurrentsInPMax(i)=PSW(MaxIdx,i);

PowersInPMax(i)=PSW(MaxIdx,i)\*PrincipalWiresV(i);

end

%% Max Power comsumtion and current at each wire

CurInPMaxeWire=zeros(size(PSW,2),1);

PowInPMaxeWire=zeros(size(PSW,2),1);

TInPMaxeWire=zeros(size(PSW,2),1);

for i=1:size(PSW,2)

CurInPMaxeWire(i)=max(PSW(:,i));

[PowInPMaxeWire(i), MaxIdx]=max(PSW(:,i)\*PrincipalWiresV(i));

TInPMaxeWire(i)=PSWT(MaxIdx,i);

end

end

## Anexo: Función para comprobar que todo el cableado incluye el bloque Wire

%% This function highlight every element with at least one pin to pin without the wire block

% Geting all Wires from model

Wires=find\_system(model,'FindAll','on','SearchDepth','1','type','line');

j=0; % Lines without Wire Block connected counter

k=0; % Lines without Wire Block connected counter

m=0; % Lines with unknown wires

% Arrays Setting Up

WithoutWire(1)=""; % Array of lines without wires

WiresOK(1)=""; % Array of wires OK.

FailedWires(1)=""; % Array of wires with no info

for i=1:size(Wires,1)

%For every line look for its origin and its destination to

if all(get\_param(Wires(i), 'Dstblockhandle')~=-1) && ...

all(get\_param(Wires(i), 'Srcblockhandle')~=-1)

Destination=string(get\_param(get\_param(Wires(i), 'Dstblockhandle'),'Name'));

Origin=string(get\_param(get\_param(Wires(i), 'Srcblockhandle'),'Name'));

if length(Destination)~=1 || length(Origin)~=1

% More than two elements connected to the same line

m=m+1;

FailedWires(m,1)=Wires(i);

elseif ((strlength(Destination)==4 && extractBefore(Destination,2)=="W" && isfinite(str2double(extractAfter(Destination,1)))) || ...

contains(Destination,'Open Circuit'))

% Case of Pin to Pin OK.

% > Destination is an Open Circuit

% > Destination is a Wire Block

k=k+1;

WiresOK(k)=Destination;

elseif (strlength(Origin)==4 && extractBefore(Origin,2)=="W" && isfinite(str2double(extractAfter(Origin,1)))) || ...

contains(Origin,'Open Circuit')

% Case of Pin to Pin OK.

% > Origin is an Open Circuit

% > Origin is a Wire Block

k=k+1;

WiresOK(k)=Origin;

elseif (string(get\_param(find\_system(model,'SearchDepth','1','Name',Destination),'BlockType'))=="SubSystem" || ...

string(get\_param(find\_system(model,'SearchDepth','1','Name',Destination),'BlockType'))=="SimscapeBlock") && ...

(string(get\_param(find\_system(model,'SearchDepth','1','Name',Origin),'BlockType'))=="SubSystem" || ...

string(get\_param(find\_system(model,'SearchDepth','1','Name',Origin),'BlockType'))=="SimscapeBlock")

% Case of Pin to Pin ERROR.

% The line is a functional element and is not conected to a wire block.

j=j+1;

WithoutWire(j,1)=Destination;

WithoutWire(j,2)=Origin;

end

else

% Case line destination or origin not found.

m=m+1;

FailedWires(m,1)=Wires(i);

end

end

WiresOK=sort(WiresOK);

for i=1:size(WithoutWire,1)

%Highlight every block with error in pin to pin

set\_param(model+"/"+WithoutWire(i,1),'BackgroundColor','w')

set\_param(model+"/"+WithoutWire(i,2),'BackgroundColor','w')

end

## Anexo: Función Troubleshooting

%% With this function broken wire tests will be use to get wire and functional element which may be the cause of test failure

% model [string]: model of the system to be troubleshooted.

%% INPUTS

% SR [3D string array]: result in each of the steps in the execution broken

% wire tests. 1D: each of the step. 2D: step number and result [1,0].

% 3D: number of execution of the test. It is consistent with WireList

% SR(idx)->WireList(idx-1). First 3D element is the execution without

% broken wire

% WireList [1D string array]: index broken wire tests.

% TestResul [1D boolean array]: Test Failed Result.

%% OUTPUTS

% PossibleBrokenWires [1D string array]: list of possible damaged wires.

% PossibelBrokenFunElements [1D string array]: list of possible damaged functional elements.

PossibleBrokenWires=WireList(find(all(SR(:,2,2:end)==TestResult)));

PossibelBrokenFunElements="";

j=0; % PossibelBrokenFunElements count

for i=1:length(PossibleBrokenWires)

Connec=get\_param(model+"/"+PossibleBrokenWires(i),'PortConnectivity');

j=j+1;

PossibelBrokenFunElements(j)=get\_param(Connec(1).DstBlock,'Name');

j=j+1;

PossibelBrokenFunElements(j)=get\_param(Connec(2).DstBlock,'Name');

end

PossibelBrokenFunElements=unique(PossibelBrokenFunElements);

# Referencias

## Bibliografía

DISEÑO LOGICO DE UN SISTEMA - diseñodesistemas\_era. (n.d.). Retrieved April 06, 2020, from https://sites.google.com/site/disenodesistemasera/diseno-logico-de-un-sistema

Georgia Institute of Technology. (2018, November 01). Retrieved April 06, 2020, from https://www.news.gatech.edu/2018/11/01/airbus-and-georgia-tech-open-center-overall-aircraft-design

Hispaviación. (n.d.). Retrieved April 07, 2020, from http://www.hispaviacion.es/

Lógica programada. (2020, February 07). Retrieved April 06, 2020, from https://es.wikipedia.org/wiki/Lógica\_programada

Requirement (glossary). (n.d.). Retrieved April 06, 2020, from https://www.sebokwiki.org/wiki/Requirement\_(glossary)

Seguridad Operacional. (2015, October 28). Retrieved April 07, 2020, from https://es.wikipedia.org/wiki/Seguridad\_Operacional

System Requirement (glossary). (n.d.). Retrieved April 06, 2020, from https://www.sebokwiki.org/wiki/System\_Requirement\_(glossary)

Using model-based systems engineering to develop the next-generation A350 XWB. (n.d.). Retrieved April 06, 2020, from https://www.plm.automation.siemens.com/global/es/our-story/customers/airbus-a350-xwb/16450/

Ventura, V. (2016, August 20). ¿Qué es la lógica programada? Retrieved April 06, 2020, from https://polaridad.es/logica-programada-programable-que-es-pld-fpga-hdl-cpld/

What is a Model? (n.d.). Retrieved April 06, 2020, from https://www.sebokwiki.org/wiki/What\_is\_a\_Model?

World MBSE Cyber System Experience Symposium: Dassault Systèmes CATIA Cyber Systems with No Magic – a New World of 3DEXPERIENCE. (2019, May 20). Retrieved April 06, 2020, from https://www.businesswire.com/news/home/20190520005258/en/World-MBSE-Cyber-System-Experience-Symposium-Dassault

(n.d.). Retrieved April 06, 2020, from https://www.boeing.com/features/innovation-quarterly/may2017/feature-technical-model-based-engineering.page