CAPÍTULO 1

INTRODUCCIÓN

1.1 TRABAJO PRECENDENTE

En este proyecto, pretendemos ir más allá de la simulación en tiempo normal de un modelo de mini-helicóptero. Anteriormente se han empleado modelos de aeronaves para la creación de leyes de control y poder ver así cómo se comportan éstas con distintos tipos de controladores mediante el uso de la simulación.

En el caso que ahora nos ocupa, pretendemos pasar de una simulación en tiempo normal (la usada hasta el momento para el cálculo de leyes de control) a una simulación en tiempo real. Más adelante daremos cuenta de la importancia que presenta hoy en día una simulación de este tipo.

1.2 ESTRUCTURA GENÉRICA DE UN SISTEMA DE CONTROL AERONÁUTICO

Un sistema de control aeronáutico genérico y el ambiente que lo rodea están estructurados según un esquema de interacción que puede ser simplificado en el modo en que se muestra en la figura 1.1 [16]:

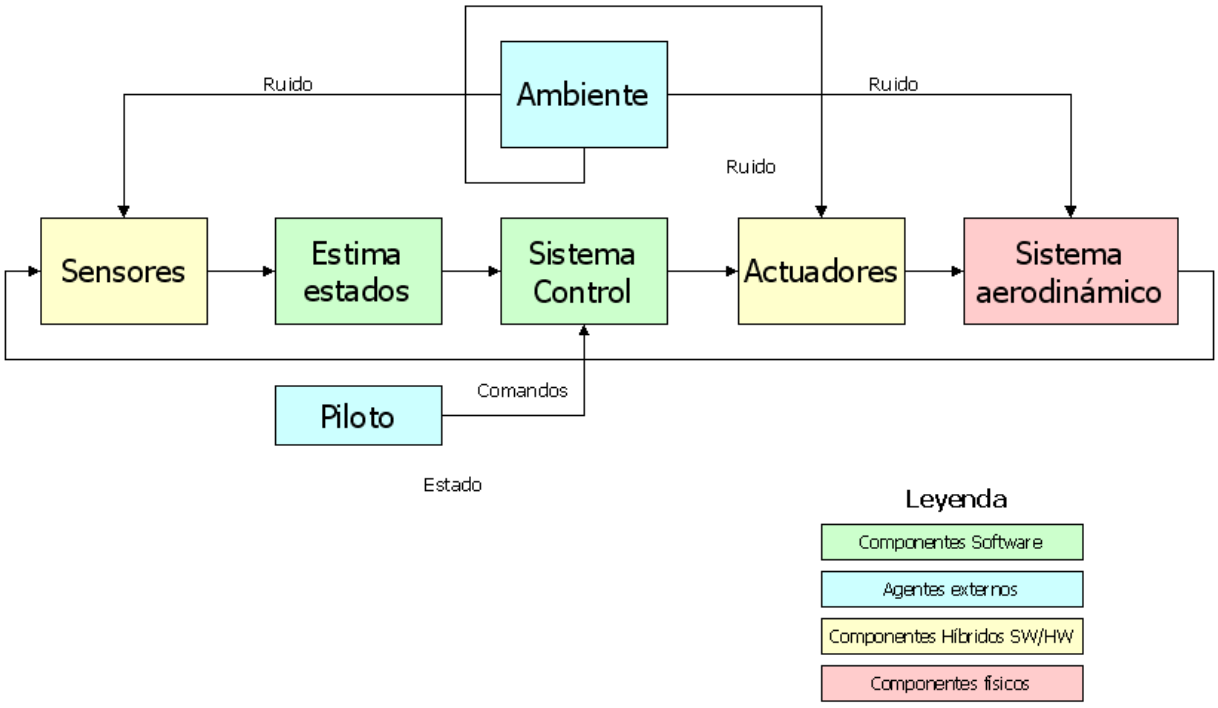


Figura 1.1: Esquema de un sistema aeronáutico controlado

Por lo tanto, un sistema aeronáutico está formado por los siguientes componentes:

- Sensores: los cuales se pueden comprender todos o en parte estos componentes, están caracterizados por una dinámica de funcionamiento, por una precisión con la cual transforman la magnitud física que miden y por la frecuencia a la cual proporcionan datos:

  ➢ADS (Air Data Systems): todos aquellos sensores que miden el estado de la atmósfera en torno a una aeronave (presión estática y dinámica, temperatura) que estarían relacionados, a través de algunas leyes de la termodinámica, con el estado de la aeronave (velocidad respecto al aire, cuota, número de Mach).

  ➢Magnetómetro: mide la dirección del campo magnético, en función de la dirección y de la actitud (posición respecto a los ángulos de Euler) de la aeronave.

  ➢GPS (Global Positioning System): sistema para el cálculo de la posición y de la velocidad actual a través de triangularización por satélite.

  ➢IMU (Intertial Measurement Unit): unidad para el cálculo de las aceleraciones lineares y las velocidades angulares de la aeronave.

- Unidad de estima del estado: cada uno de los sensores que alimenta el bloque de estima del estado proporciona una medida de una parte del estado de la aeronave, con diferentes precisiones y frecuencia de medida. Un algoritmo de estima del estado, basado normalmente en un filtro de Kalman, permite calcular el estado del sistema aeronáutico y de los mismos sensores (por ejemplo, para las unidades inerciales es frecuentemente necesario medir las características de *bias* y de *drift* que influyen fuertemente en la medida de las unidades inerciales); esta unidad es normalmente un bloque independiente, cuyas entradas son los sensores de la aeronave y sus salidas son, a su vez, las entradas al sistema de navegación y control.

- El sistema de control: frecuentemente viene representado como un único bloque; dicho sistema en realidad está constituido también por un sistema de navegación (que define la ruta de la aeronave, en caso de elegir la opción de piloto automático) y por un sistema de control propiamente dicho (que tiene como misión estabilizar la aeronave para conferirle las características de maniobrabilidad requeridas por la especificación de diseño); este último bloque se construye normalmente en función de las características aerodinámicas de la aeronave y de las especificaciones técnicas según las prestaciones que haya que alcanzar.

- Los actuadores: comandados por el sistema de control y capaces de modificar de un modo eficaz el estado del sistema aerodinámico, son considerados a menudo como sistemas completamente mecánicos caracterizados por una dinámica más o menos significativa, cuya interacción con el ambiente circundante puede ser considerada despreciable.

- El sistema aerodinámico: suma de las características aerodinámicas de la aeronave, diferentes para cada modelo (que describen la interacción de la aeronave con el aire que la rodea), y de las leyes dinámicas y cinemáticas que regulan el movimiento de cualquier cuerpo rígido. Las características aerodinámicas vienen a ser diseñadas de manera que sea posible obtener las especificaciones de diseño requeridas en términos de maniobrabilidad, carga útil transportada y autonomía.

- El ambiente: condiciona fuertemente el movimiento de una aeronave generando ruidos de naturaleza aerodinámica, mecánica y eléctrica, cuyas características se pueden modelar (si son significativas) a través de modelos estándar; el sistema de control se diseña también para soportar un nivel de ruido definido por especificaciones.

- El piloto: hemos de considerarlo del mismo modo que un "ruido"; el piloto impone las características del movimiento modificando el estado de equilibrio dinámico de manera continua. El piloto también puede ser caracterizado a través de modelos estándar de interacción entre piloto y aeronave.

1.3 MOTIVACIONES DE LA SIMULACIÓN DE UNO O MÁS COMPONENTES DE UN SISTEMA AERONÁUTICO

Con los rápidos avances en sistemas de aviónica, avanzados controles y displays de cuadros de control de las cabina de pilotaje, tecnología fly-by-wire y demás, la necesidad de pasar de forma conveniente del concepto a la certificación es un requisito fundamental a la hora de desarrollar con éxito un proyecto de aeronave.

El uso de la simulación para desarrollar un producto superior aeroespacial o de defensa ha sido adoptado en la industria. La simulación ha permitido fabricar motores más silenciosos para producir un mayor empuje con menos consumo de combustible. La simulación ha hecho posible además la reducción del peso de las aeronaves, ha mejorado la eficiencia aerodinámica, un rango extenso de aeronaves, y como resultado una mayor carga de pasajeros y de compartimentos de carga. Los proyectos de desarrollo de aeronaves dependientes de simulación llevados a cabo con éxito alcanzan un mayor grado de perfeccionamiento, por esto, los diseñadores miran atrás para preguntarse "¿Cómo podríamos hacer mejor las cosas?". Una respuesta muy común a esta pregunta es: "Con una simulación más precisa" [1].

1.4 EL PROCESO DE DESARROLLO POR PASOS

El *simulation-centric process* aproxima sistemas complejos con los siguientes procesos por etapas:

1. Diseño preliminar

2. Diseño basado en simulación

3. Cosimulación

4. Early integration

5. Late integration

La figura 1.2 ilustra el flujo del proceso. Cada etapa en el proceso tiene una o varias entradas y una o varias salidas. Se hace uso de varias herramientas en cada etapa para poder trabajar con las entradas y desarrollar las salidas [2].

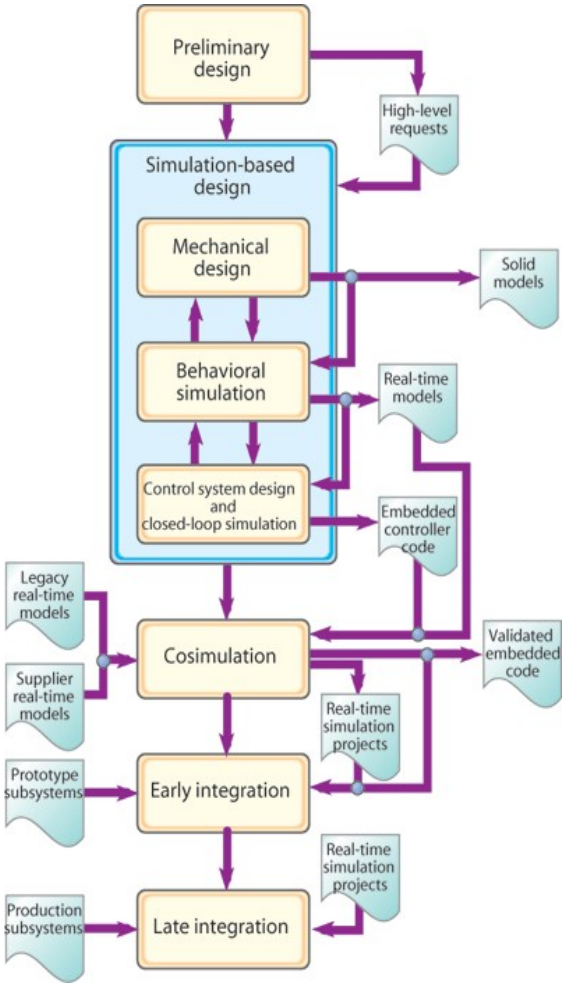


Figura 1.2: Diagrama de flujo del proceso

1.4.1 Etapa 1: Diseño preliminar

La etapa de diseño preliminar en el *simulation-centric process* es parecida a la mayoría del resto de procesos de desarrollo. Durante el diseño preliminar, el equipo define los requisitos de alto nivel para el sistema. Dichos requisitos podrían incluir el peso del sistema, rango, velocidad, eficiencia, etc… Después, el equipo descompone el sistema en subsistemas y especifica, a su vez, los requisitos de alto nivel para cada uno de ellos. Finalmente, definen las interfaces entre subsistemas. La etapa de diseño preliminar genera una especificación de alto nivel, usada para guiar posteriormente las labores en la siguiente etapa del proceso [2].

En lo que se refiere a ciclos de desarrollo de un sistema aeronáutico complejo, es muy común simular el sistema completo operando en su entorno deseado. Dicha simulación suele ser una aplicación desarrollada en modo de tiempo no-real usando una herramienta de simulación del sistema dinámico como podría ser Simulink. Esta simulación en tiempo *no real* puede servir como base para una simulación en tiempo real. A veces es necesario realizar simplificaciones y optimizaciones en los modelos contenidos en la simulación para así adaptarlos para su uso en la simulación en tiempo real. En muchos proyectos, no es necesaria ningún tipo de modificación para que puedan ejecutarse en tiempo real [1].

1.4.2 Etapa 2: Diseño basado en la simulación

La segunda etapa es el diseño basado en la simulación. El objetivo del diseño basado en la simulación es el de permitir rápidas iteraciones entre el diseño mecánico, la simulación del comportamiento, el diseño de los sistemas de control, y la simulación en bucle cerrado (*closed-loop*).

Primero, se desarrollan modelos sólidos del comportamiento mecánico usando herramientas CAD. Estos modelos sólidos se convierten entonces en modelos conductuales usando herramientas de simulación conductual (*behavioral-simulation*). Las nuevas características de estas herramientas de modelado permiten que una gran parte de esta conversión sea automatizada. Las simulaciones conductuales evalúan el diseño mecánico.

Finalmente, los modelos de simulación conductual son usados en simulaciones en bucle cerrado (*closed-loop*) para asistir el diseño y evaluación de sistemas de control.

El diseño basado en simulación requiere un gran esfuerzo colaborativo del equipo, involucrando diseñadores mecánicos, ingenieros en simulación conductual y diseñadores de sistemas de control.

Los miembros del equipo comparten modelos sólidos, modelos en tiempo real e informes de problemas. Las salidas de esta etapa del proceso incluyen modelos sólidos que son entregados a los fabricantes, modelos en tiempo real usados en cosimulación e integración, y código de control *embedded* usado para la *early integration* y enviado a los proveedores de *embedded systems*. Las herramientas usadas para ayudar a mejorar la eficiencia del diseño basado en la simulación están evolucionando rápidamente. La Tabla 1.1 hace un listado de las herramientas para el diseño basado en simulación más conocidas [2].

Un valor de las herramientas en el proceso depende tanto de su habilidad para importar desde la etapa anterior, como de la de asistir el desarrollo de salidas para la siguiente etapa.

Cad Tools	Behavioral Modeling	Control System Design
CATIA	ADMAS	Simulink
Pro/Engineer	Dymola	Statemate
SolidWorks	AMESim	Rhapsody
Others	Simulink	Hand Code
	Easy5	

Tabla 1.1: Simulation-based design

1.4.2.1 El software para la simulación

Hoy en día disponemos en ingeniería de una aparentemente selección interminable de herramientas. Diferentes herramientas demuestran una precisión superior en el diseño y simulación de distintas piezas de una aeronave.

Entre los métodos de simulación disponibles para sistemas de diseño de aeronaves tenemos:

- modelado hidráulico para sistemas de aviones hidráulicos
- modelado de dinámicas de fluidos para modelado y diseño aerodinámicos
- modelado de sistemas discretos para el diseño de sistemas eléctricos

- modelado de sistemas de control para aviónica y diseño de controladores FADEC
- termodinámicas para simulación de motores reactores de combustión
- simulación “multibody” para diseño y simulación mecánica

Usando una única herramienta para simular la aeronave al completo no sólo lleva a reducir la precisión sino que también limita al manufacturador a un único proveedor y puede dar lugar a costes cada vez más elevados. Los equipos de diseño preliminar están usando media docena de herramientas de simulación para el desarrollo de aeronaves de próxima generación.

Algunas de las herramientas de simulación más conocidas son [1]:

- Adams
- AMESim
- Dynasim
- Easy5
- Matlab/Simulink/Stateflow
- MatrixX/SystemBuild
- Rhapsody
- Statemate

Un programa normal de desarrollo de aeronaves usará un surtido de estas herramientas.

Debido a que las simulaciones de sistemas complejos requieren gran cantidad de algoritmos numéricos muy sofisticados, se han desarrollado una serie de herramientas software especializadas para simplificar la tarea [1]:

- Simulink es una herramienta añadida a Matlab que permite la simulación de un sistema dinámico en un entorno gráfico de diagramas de bloques. Para desarrollar una simulación en Simulink sólo hay que arrastrar bloques desde una paleta a un área de dibujo y conectar los bloques entre sí con líneas que representan los flujos de señal.

- Stateflow es una herramienta añadida a Simulink que permite la implementación de modelos de máquinas de estados finitos. En un proyecto de helicóptero, un modelo Stateflow debería implementar la lógica de selección de modo del helicóptero.

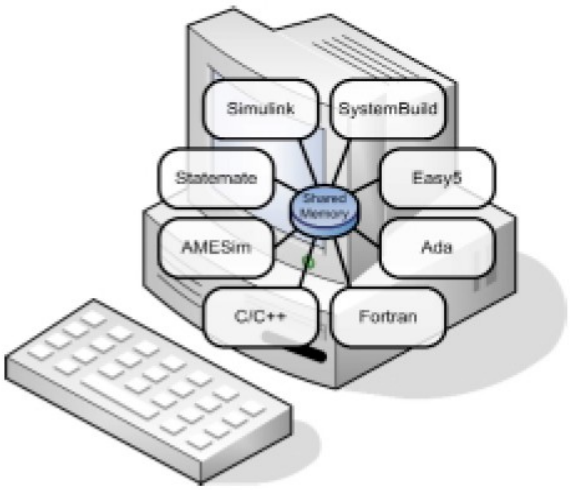
- xPC Target habilita la ejecución de una simulación en un PC que funciona como un controlador en tiempo real. Proporciona un kernel multitarea en tiempo real para ser usado en un procesador *embedded* con un número limitado de recursos hardware. En un proyecto, *xPC Target* puede ser usado para generar y ejecutar el código en tiempo real para el controlador del helicóptero en un PC independiente, el cual haría de controlador *"embedded"*.

1.4.3 Etapa 3: Test de integración con *Cosimulación*

Una vez que la simulación piece-wise[1] independiente de la aeronave ha sido completada, el siguiente paso es unir todas las piezas.

Cada pieza debe comportarse como era de esperar por sí sola, pero podría ser que no se comportara de igual modo cuando se combina con otras. Para hacer cosas más complejas, se suelen usar modelos de simulación basados en código *(legacy code-based simulation models)* de programas previamente desarrollados para reducir esfuerzos de diseño. Dichos modelos de simulación basados en código suelen ser del tipo:

- Ada
- Fortran
- C/C++



1 Simulación de cada uno de los componentes por separado.

La *cosimulación* es la técnica mediante la cual se toman numerosos componentes simulados, diseñados usando uno o más paquetes de modelado, y/o escribiendo en uno o más lenguajes de código (Ada, C, C++, Fortran), y desarrollando una simulación combinada de estos múltiples componentes.

El objetivo de la cosimulación es evaluar cómo de bien funcionan todos los componentes juntos. Por ejemplo, un proyecto de cosimulación de una aeronave completa podría implicar lo siguiente:

- simulación del motor usando un modelo Fortran
- simulación FADEC (*Full Authority Digital Engine* Control) usando un modelo ADA
- aerodinámicas usando un modelo Simulink
- hidráulicas de la aeronave usando un modelo Easy5
- simulación del sistema eléctrico usando Statemate
- simulación del tren de aterrizaje usando Dymola
- comunicación Databus modelada en C++

Durante la cosimulación, todos los modelos son ejecutados en modo *"lock-step*" sincronizado usando una plataforma de cosimulación. Las herramientas de cosimulación son ejecutadas en un entorno PC o Unix. La plataforma de simulación ejecuta los modelos de simulación, proporciona resultados y análisis, test de automatización, visualización y control.

La interacción de cada componente simulado es analizada y comparada con los requisitos de simulación. Los errores de diseño se detectan antes en el ciclo de desarrollo y pueden ser corregidos rápidamente con un gasto mínimo.

La salida de un proyecto de cosimulación llevado a cabo con éxito es una colección de modelos de simulación testeados exhaustivamente. Estos modelos pasan después a la siguiente fase de desarrollo donde conducen el comportamiento a unidades de desarrollo rápido.

En definitiva, la cosimulación es una nueva herramienta para muchas compañías de automatización, aeroespacio y defensa. Los jefes de programación usan un cálculo de la tasa de carga (*burden-rate*) como medida del coste de usar herramientas,

máquinas e instalaciones. Una tasa de carga alta sugiere que la herramienta es cara para operar con ella. Los computadores de simulación de tiempo real usados durante la *early* y la *late integration* son muy costosos y por lo tanto poseen una alta tasa de carga. El propósito principal de la cosimulación es hacer operar todos los subsistemas al completo y validar el código del *embedded system* en una plataforma con baja tasa de carga (un PC Windows o una workstation Unix) antes de empezar a desarrollar el prototipo.

Los modelos de simulación usados durante la cosimulación provendrán tanto de etapas del proceso de diseño basados en simulación como de proveedores o de proyectos de diseño previos. La cosimulación completa incluirá normalmente varios proyectos más pequeños de cosimulaciones de subsistemas. Cada proyecto será desarrollado por los propietarios de cada subsistema. Un subsistema debería pertenecer a un equipo de desarrolladores de *embedded systems*, diseñadores mecánicos, o ambos.

El desarrollo de un proyecto de cosimulación empieza cuando el diseño basado en simulación se estabiliza. La ejecución y análisis de la cosimulación tienen lugar en paralelo con el diseño basado en simulación y la *early integration.* Los errores de diseño obtenidos usando la cosimulación, retroalimentarán la etapa de diseño basado en simulación y darán lugar a una iteración de diseño. La figura 1.3 ilustra el modo en que la simulación de subsistema y el código *embedded* dan paso a la cosimulación:

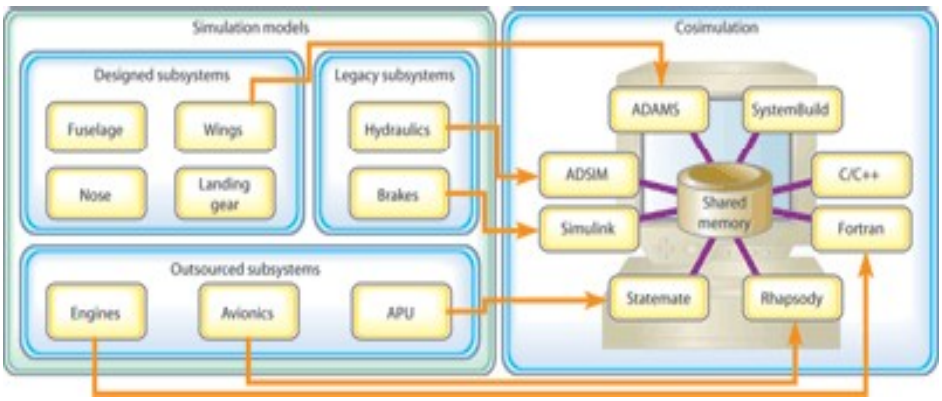


Figura 1.3: Paso de modelos de subsistemas y código de *embedded system* a cosimulación

El propósito secundario de la cosimulación es crear una plataforma de bajo coste (normalmente una estación de trabajo Windows o Unix) para la parte más dura del trabajo de la integración del sistema. Esto implica preparar los modelos para una simulación en tiempo real y tests de desarrollo reutilizables durante las etapas de *early* y *late* integración. Las herramientas usadas para la cosimulación deberían ser idealmente las mismas herramientas usadas en las dos etapas de integración. Usando las mismas herramientas para la cosimulación y la integración ahorramos en esfuerzo a la hora de configurar los modelos de cosimulación, documentar las interfaces, etc...

De esta manera, la creación de los procedimientos de testeo se realizan en la primera parte del proceso de desarrollo, evitando así el tener que repetirlos en los test de integración. Las herramientas de cosimulación han de soportar la ejecución sincronizada de modelos provenientes de numerosos lenguajes de modelado. La herramienta de cosimulación más común que podemos encontrar en la industria aeroespacial o de defensa es *ADI's Advantage/GP* usada por Boeing, Rolls-Royce, la marina de los Estados Unidos, Gulfstream, y otros más. *ADI's Advantage/GP* es conocida porque sus ficheros de proyecto de cosimulación se pueden mandar a la etapa de integración del proceso y se pueden ejecutar en procesadores de simulación *ADI's PC-based* y procesadores de simulación *VME-based real-time*.

Las salidas de la etapa de cosimulación incluyen un código de *embedded system* validado que es enviado al proveedor del *embedded system*, proyectos usados para la simulación en tiempo real, y los procedimientos de testeo que serán reutilizados durante la integración [1].

1.4.4 Etapa 4: Early Integration en laboratorios de test

Como los componentes reales de aeronaves están disponibles, dichos componentes son añadidos al laboratorio de tests de integración quitando el RDU y conectando el componente real usando las interfaces eléctricas y buses de datos previamente añadidos y testeados.

Los sistemas de aviónica de aeronaves, controles de vuelo, y los displays de los cuadros de control de las cabinas de pilotaje, son los candidatos primordiales para el testeado en laboratorios de integración. Estos sistemas son altamente complejos, altamente interconectables y requieren una gran interacción humana. Una gran cantidad de horas de testeado *man-in-the-loop* son necesarias con estos sistemas para conseguir certificarlos eventualmente. Estos sistemas son añadidos a los otros componentes de la aeronave en la integración por medio del laboratorio de test y son conectados al resto de la aeronave usando buses de datos, como por ejemplo ARINC429, así como otras interfaces eléctricas. Los pilotos se sientan al mando de los controles y desarrollan una simulación de vuelo altamente precisa. Los pilotos buscan anomalías en las pantallas LCD, evalúan el comportamiento de los controles avanzados de vuelo, y evalúan el manejo global de la aeronave. Este conjunto de tests tiene lugar antes de que la línea de ensamblado de la producción de la aeronave sea puesta en funcionamiento. Los problemas son detectados y asistidos antes en el ciclo de desarrollo.

Los laboratorios de integración son ideales para el desarrollo de nuevas versiones de aeronaves existentes. Un laboratorio de test de integración ya existente es utilizado y adaptado para acoger nuevos sistemas. Idealmente, los nuevos sistemas son añadidos de manera que puedan ser fácilmente intercambiables por los viejos

sistemas. Esta aproximación de *rápido intercambio* permite al laboratorio de integración testear rápidamente cada versión de la aeronave en cualesquiera condiciones de vuelo. Nuevos sistemas son añadidos a un simulador de integración previamente verificado y el proceso de testeado de desarrollo permite minimizar la mayoría de tests de vuelo caros.

La *early integration* es el primer paso hacia la simulación en tiempo real y es la primera vez que el código del *embedded system* es ejecutado en tiempo real. La etapa de *early integration* crea una facilidad de testeado donde:

1. Los efectos de desarrollo de las interfaces de redes y la arquitectura de redes pueden ser testeados

2. Subsistemas prototipo pueden ser testeados con un sistema completo de complejidad *in-the-loop*

3. Se pueden desarrollar actividades de selección de sensores (por ejemplo, precio vs efectos de desarrollo sobre el sistema completo)

4. Es posible verificar en la fase inicial del proyecto la interacción entre el piloto y el sistema completo

El laboratorio de *early integration* está constituido por simuladores sincronizados de tiempo real. Históricamente, los laboratorios de integración usarían una computadora con un gran multiprocesador de simulación en tiempo real para ejecutar todos los modelos de simulación y el código *embedded*. La experiencia nos ha enseñado que usar muchos simuladores pequeños sincronizados reduce la complejidad de la simulación en tiempo real y permite a un mayor número de ingenieros tener acceso a una computadora de tiempo real para su desarrollo inicial [2].

1.4.4.1 La simulación Hardware In The Loop (HIL)

Los ingenieros están constantemente buscando formas de acortar el ciclo de desarrollo del nuevo producto. Una forma de conseguir este objetivo es desarrollar el hardware y el software en paralelo. Tradicionalmente, este enfoque implica tener que separar los grupos de desarrollo hardware y software que desarrollan su trabajo simultánea e independientemente. Cuando se obtiene un prototipo hardware y contamos con una parte sustancial de código *embedded*, el hardware y el software se combinan en una fase de integración de sistema y comienza el testeado.

Demasiado frecuentemente, surgen serios problemas durante la fase de integración del sistema, problemas que requieren un nuevo y significante trabajo en

cuanto a hardware y software se refiere. Los proyectos pueden paralizarse durante la integración del sistema, estancados con crecientes listas de problemas, disparando los costes, y aumentando retrasos en las fechas de entrega de los proyectos. A veces, como resultado, los proyectos son cancelados. Claramente, necesitamos una mejor forma de encargarnos de estas cuestiones.

Un enfoque que ha resultado ser efectivo es la simulación *hardware-in-the-loop* (HIL). Esta técnica permite el testeo de software *embedded* en una etapa mucho anterior del ciclo de desarrollo. Desde el momento en el cual empieza la integración del sistema, el software *embedded* ha sido testeado mucho más minuciosamente de lo que lo habría sido según el enfoque tradicional. Cuanto antes identifiquemos los problemas, menos nos costará darles una solución.

Algunos ajustes de las ganancias del controlador, que son diseñados usando la simulación HIL, son siempre necesarios para obtener un rendimiento aceptable del sistema para todos los modos de operación. La simulación HIL no encajaba mucho con el comportamiento de los sistemas actuales porque la simulación de los sistemas aeronáuticos se simplifica en cierta manera y las propiedades de masa del sistema usadas en la simulación a menudo no coinciden con las de los sistemas actuales.

Es necesario siempre simplificar a algún grado cuando estamos desarrollando una simulación porque no es posible modelar a la perfección cada factor influyente en el comportamiento de un sistema real. Las diferencias entre la simulación y el sistema actual han resultado ser menores y el ajuste de parámetros requerido para el software *embedded* era sólo un procedimiento sencillo.

Las técnicas de simulación HIL permiten un testeado minucioso del software *embedded* al principio del proceso de desarrollo y reducen los riesgos que conlleva la ejecución de software, que aún no ha sido testeado, sobre prototipos hardware costosos. Una correcta aplicación de técnicas de simulación HIL da lugar a productos de una calidad mayor y desarrollados en menos tiempo que con las técnicas de desarrollo tradicional [3].

1.4.4.2 Rapid development unit integration testing

Las unidades de desarrollo rápido (RDUs) están ganando popularidad en el desarrollo de vehículos aéreos y de defensa.

El concepto consiste en usar varias computadoras “pequeñas” de simulación en tiempo real, cada una comportándose como un componente de la aeronave, conectadas usando buses de datos reales e interfaces eléctricas.

Las computadoras de simulación en tiempo real basadas en ADI's VME y las

computadoras de simulación expansible en tiempo real rtX PC-based se suelen usar a menudo como *targets* RDU. La simulación en tiempo real *multi-computer* proporciona el primer paso hacia el simulador de testeado de integración completo. Un panel e control simulado se podría añadir con displays y controles, y así podríamos comenzar a simular los primeros elementos del factor humano. El objetivo de los tests de integración RDU es asistir en el diseño y prueba de las interfaces y sensores del sistema.

Cada computadora de tiempo real ejecuta un modelo que fue desarrollado durante una simulación pura y testeado durante la cosimulación. La adición de interfaces eléctricas reales y de una comunicación por buses de datos reales introduce una nueva ley de comportamiento. El desarrollo de estas interfaces eléctricas y estos buses de datos tiene un mayor impacto sobre el desarrollo total de la aeronave. El objetivo de los tests de integración RDU es el de proporcionar una plataforma donde la comunicación interfaz podría ser evaluada tan bien como el impacto de esta comunicación sobre cada sistema de la aeronave. Las pruebas de desarrollo de estas interfaces eléctricas se pueden dividir en desarrollo de redes y selección de sensores [1].

1.4.4.3 Network development

El uso de redes de vehículos y aeronaves continúa creciendo. Implementando redes como ARINC 429, MIL-STD-1553, CAN y AFDX en lugar de sistemas de cableado directo y sensores muy pequeños reduce la cantidad de cableado y eso a su vez reduce el peso de la aeronave y el número de puntos de fallo en el sistema eléctrico.

Cuando se usan redes de aeronaves el reto que se nos presenta es asegurar que la topología de la red pueda funcionar y rendir como esperamos cuando se experimenten los peores casos de cargas de tráfico.

El test de integración RDU está diseñado específicamente para resolver los problemas de diseño de la red de la aeronave.

Las RDUs están conectadas a redes de aeronaves reales y se comunican unas con otras, tal y como sería en el sistema final de la aeronave. Los modelos de simulación, comportándose de un modo real, conducen el tráfico de la red. Llevando el simulador basado en RDU a través de tests reales como por ejemplo despegues simulados, aterrizajes y otras maniobras, se identifican periodos de alto tráfico en la red. Los datos de simulación son registrados y analizados para comparar diferentes tipologías de redes.

Además, el impacto de la comunicación de redes puede ser evaluado en términos

de impactos de desarrollo sobre los sistemas de las aeronaves [1].

1.4.4.4 Selección de sensores

La siguiente aplicación importante del testeo de integración RDU es la selección de sensores. Los distintos tipos de sensores poseen puntos fuertes y débiles según nos movamos con precisión a lo largo de diferentes rangos, diferentes costes y aplicaciones específicas.

Usando el testeo de integración RDU, se puede probar una selección de sensores "candidatos potenciales" en el simulador de aeronaves RDU al principio del ciclo de desarrollo. El comportamiento del sensor puede ser simulado en tiempo real usando datos de sensores y de esta manera se podría medir el efecto de un sensor particular en el mecanismo de una aeronave completa.

Alternativamente, podríamos conectar sensores reales a los RDUs usando una interfaz eléctrica apropiada. En este caso, los sensores reales son evaluados basándose en cómo de bien se ajustan a la aplicación. Los tests de vuelo RDU proporcionan una cantidad increíble de información sobre las características finales de la aeronave al principio del desarrollo [1].

1.4.5 Etapa 5: *Late integration*

La quinta y última etapa del *simulation-centric process* (proceso de desarrollo basado en la simulación) es el *testeado late-integration.* Una vez que la *early integration* ha sido completada, la *facility* de la *early integration* es transformada, subsistema a subsistema, en la *facility* de la *late integration*. Como los subsistemas proceden de proveedores y equipos de desarrollo, son programados como actividades de *late-integration*. Los subsistemas se van añadiendo uno a uno para aislar la fuente de cualquier problema. Los proveedores deberían programar el reparto de varias generaciones de prototipos para un período de aproximadamente un año. Las primeras generaciones son modulares y tienen un fácil acceso a los circuitos para pruebas. Las tarjetas intercambiables nos permiten evaluar con bastante precisión diferentes microprocesadores y circuitos de sensores. Los prototipos de *later-integration* empiezan a parecerse a los del sistema de producción final. Los circuitos pasan a ser más densos y el acceso a los mismos pasa a ser mas complicado. Las fuentes de muchos problemas de software *embedded* son los modernos displays de los cuadros de control de las cabinas de pilotaje y su interacción con la aviónica y otras computadoras de control. El laboratorio de *late integration* pasará a través de docenas, incluso centenas, de actualizaciones de software antes del primer test sobre un nuevo sistema final. Tests de pilotos emplearán miles de horas evaluando y testeando complejos controles y displays en la *facility* de la *late-integration*. Las herramientas de la *late-integration* son a menudo muy caras porque incluyen millones

de dólares de equipamiento de prototipo. Los altos costes de los laboratorios de *late-integration* requieren que la mayor parte de los tests de desarrollo se realicen mejor durante la *early integration* que durante la *late integration* [2].

1.5 EL CASO QUE NOS OCUPA

En los apartados anteriores hemos dejado constancia de la importancia que tiene la simulación en tiempo real. Se trata, básicamente, de ahorrar en tiempo, costes, mano de obra y obtener resultados cada vez mejores y con mayor rapidez.

En el caso que nos ocupa, disponemos de un modelo de helicóptero, y de lo que se tratará será de obtener una simulación en tiempo real, en la cual poder, por ejemplo, testear distintas baterías de controladores y poder hacer un análisis de los resultados a través de distintas pruebas, pudiendo así elegir las leyes de control que mejor se adapten a los requisitos exigidos. Todo esto, como no, tendrá lugar antes de pasar a realizar experimentos costosos, con lo cual ahorraremos en recursos de todo tipo.

En nuestro caso, trataremos de realizar una simulación en tiempo real de un modelo de mini-helicóptero (modelo basado en un conjunto de bloques de Simulink), al cual aplicaremos unas entradas (provenientes de un joystick) a través de una interfaz proporcionada por Aerosim, y a su vez, iremos obteniendo unas salidas, las cuales harán a su vez de entradas (a través de otro bloque de interfaz de Aerosim) al Microsoft Flight Simulator (que hará de interfaz gráfica, a través de la cual podremos visualizar y analizar los resultados obtenidos).

Por último, y antes de proseguir con el análisis del sistema que nos ocupa, hacer hincapié en la importancia de la simulación que queremos realizar: queremos ver cómo se comportaría nuestro helicóptero ante diversas leyes de control (o simplemente ante el ajuste o variación de cualquier otro parámetro de diseño), por lo que aquí nos concentraremos en tratar de disponer de un modelo lo más “real” posible para realizar dichas pruebas, es decir, queremos un sistema que se esté ejecutando en tiempo real, para así poder simular lo que sería el comportamiento real de nuestro helicóptero.