# MODELOS DE ANALISIS ORIENTADO A OBJETOS APLICADOS EN EL DOMINIO AERONAUTICO

Modelos y Patrones de Diseño

Ing. Daniel S. Monserrat

Director: Dr. Gustavo Rossi

Codirector: Ing. Victor Caballini

Tesis en el campo de la Ingeniería de Software

para el grado de Magíster en Ingeniería de Software

Universidad Nacional de La Plata

Facultad de Informática

Secretaría de Postgrado

2005

## Resumen

La simulación en el campo aeronáutico constituye una herramienta importante, a tal punto que el primer simulador, como se conocen actualmente, ha sido desarrollado en el dominio aeronáutico. La importancia como herramienta radica en que tanto en el ámbito Universitario como en la industria Aeronáutica es necesario contrastar los resultados teóricos con la práctica; pero lamentablemente, en los aspectos relacionados con la aerodinámica aplicada y la estabilidad y control, la única posibilidad sería la de construir los prototipos de aeronaves en estudio. Es aquí donde la simulación se constituye una herramienta invaluable en la evaluación y determinación de los comportamientos reales sin la necesidad de arriesgar vidas y aeronaves. Para dar un ejemplo que aclare estos conceptos, podemos mencionar la situación de detención de motores en el despegue, que sería riesgosa o catastrófica, en una aeronave real, pero no pasará de un momento de tensión si se simulara correctamente en un dispositivo en tierra.

Es importante considerar también que las aeronaves requieren permanentes controles y verificación, que en algunos casos cambia las recomendaciones operativas aún después de 30 años de producidas, generándose constantes actualizaciones en los modelos, generados por principios de seguridad y la previsibilidad del comportamiento dinámico de la misma.

La presente tesis investiga las arquitecturas de software teóricas y utilizadas en simuladores, como así también trabajos sobre simulación en el dominio aeronáutico. El objetivo es mejorar el modelado mediante la aplicación de Análisis Orientado a Objetos intentando identificar patrones de diseño aplicables, al igual de documentar los que surjan del estudio de este dominio.

El enfoque hacia los patrones de diseño obedece a sus objetivos de origen: esta tesis intenta ser una contribución a la captura de experiencia en el dominio y la documentación de decisiones de diseño.

Por último los principios directores del presente trabajo se enumeran como el maximizar el reuso, mejorar los simuladores existentes y ser una base sobre la cual desarrollar nuevos simuladores.

## Reconocimientos

Quisiera agradecer a Gustavo Rossi por su apoyo y guía en la realización de este trabajo. A Víctor Caballini por su apoyo y continuas enseñanzas en el campo aeronáutico.

Gracias a los miembros del Grupo de Simulación Dinámica del Vuelo de la Universidad Tecnológica Nacional, Facultad Regional Haedo, con quienes compartimos el trabajo y entusiasmo en este campo.

Por último, mucha gente se ha involucrado en este trabajo y en estas líneas va mi agradecimiento a todos ellos.

## Dedicación

A mi familia, que con su apoyo y aliento han hecho posible esta tesis.

# Tabla de Contenido

# Tabla de Figuras

# Capítulo 1 Introducción

Un sistema de simulación en aeronáutica es el objeto de estudio que permite lograr un mejor entendimiento de la aeronave real o que se está diseñando. Este estudio puede ser conducido en una amplia variedad de áreas como pueden ser estudios de ingeniería, análisis de cualidades de la aeronave; diseño o entrenamiento de tripulaciones. El sistema de simulación, en general, se encuentra constituido por el modelo matemático, el hardware de simulación, el sistema visual y opcionalmente el sistema generador de movimiento; siendo el comportamiento de este conjunto lo suficientemente similar a la aeronave bajo estudio. La implementación del modelo matemático se materializa en el software que forma el simulador y constituye uno de los sistemas de software más complejos que existen [BASS03] tal como se podría desprender del hecho que se está simulando uno de los sistemas físicos más complejos que existen. Otro hecho indiscutible, es que el avance tecnológico provoca que los sistemas de las aeronaves sean cada vez más complejos y numerosos provocando el aumento en complejidad del software de simulación, el cual cada vez se hace más **complejo de mantener y evolucionar**.

Considerando esta problemática tenemos que si bien hace tiempo que son conocidos los beneficios de una estructura modular y estandarizada, e inclusive respecto a costos (principalmente cuando se deben simular diferentes tipos de aeronaves), la realidad nos indica que el desarrollo de este campo con la aplicación de tecnologías basadas en objetos es relativamente nuevo y no existen demasiados trabajos al respecto divulgados. Es por esto que la presente tesis plantea el uso de modelos orientado a objetos y en especial el uso de patrones de diseño para resolver estos problemas, aportando un enfoque nuevo al desarrollo de simuladores, enfocado en tres objetivos:

- ✓ Maximizar el Reuso
- ✓ Facilitar el Mantenimiento
- ✓ Posibilitar la Evolución

Otro aporte totalmente nuevo es que el desarrollo de estos modelos se harán a partir de estructuras conocidas utilizadas actualmente en simulación

como es el AVSM (ADA procedural) para constituir un planteo en base a la evolución de planteos anteriores que han dado resultados exitosos. Otro aporte totalmente original que se encontrará en el desarrollo de este trabajo es el planteo y documentación modelos y formas de implementación como también patrones de diseño específicos del dominio.

En la presente tesis se plantea la aplicación de patrones de diseño y modelos orientados a objetos para lograr la simulación del vuelo de aeronaves de forma que sea adaptable, fácil de mantener y permita al ingeniero Aeronáutico concentrarse en los problemas del diseño de aeronave bajo estudio. La aplicación de OOA nos permite una codificación modular, esconder los detalles de implementación y crear una estructura de código jerárquica.

Las contribuciones más importantes de esta tesis son:

- ✓ Mostrar la aplicación de patrones de diseño en simulación en el dominio aeronáutico.
- ✓ Plantear una arquitectura orientada a objetos para simuladores de aeronaves, con descripción de sus componentes y mecanismos de comunicación.
- ✓ Caracterización de los objetos que forman una aeronave y su medio ambiente.
- ✓ Aplicación sobre soluciones existentes en el dominio, de una concepción actualizada basada en el Diseño Orientado a Objetos.

Comenzaremos en el capítulo 2, con una introducción conceptual sobre simulación y el problema de la simulación de aeronaves, para ubicar al lector en el contexto que nos ocupa. En el capítulo 3 haremos una breve presentación a los patrones de diseño, su definición y nomenclatura ya que constituyen una de las bases teóricas en que se basa esta tesis. Continuando con la base conceptual que requeriremos en el desarrollo, en el capítulo 4 realizaremos el estudio de trabajos sobre simulación en el dominio aeronáutico y presentaremos modelos estructurales utilizados en la simulación del vuelo de aeronaves, haciendo hincapié en el AVSM desarrollado por el SEI ya que constituirá otra de las bases teóricas para el presente trabajo.

Con estos cuatro primeros capítulos tendremos el marco teórico, obtenido de la revisión bibliográfica existente, para el desarrollo de la tesis.

En el capítulo 5 plantearemos el patrón de diseño que representa el modelo estructural desde una perspectiva fresca basada en la orientación a objetos y con la cual pretendemos documentar este modelo para su transmisión y estandarización. En el capítulo 6 presentaremos los modelos enfocados a representar los subsistemas de la aeronave, haciendo referencia a los patrones de diseño generales que se irán adaptando a nuestro dominio. En el capítulo 7 estudiaremos y extenderemos los modelos para abarcar la interfaz gráfica (normalmente excluida por el AVSM y demás modelos). En el capítulo 8 presentaremos el modelo ambiental siempre desde el punto de vista de los patrones de diseño a aplicar. Finalmente en el capítulo 9 presentaremos el análisis y las conclusiones obtenidas de los modelos planteados en esta tesis.

# Capítulo 2 Conceptos Básicos

*"OO Software engineering methodologies are the greatest thing since sliced bread. In computer related terms object oriented techniques are the most important development since the introduction of structured techniques during the 1970's and 1980's"* [YOU94]

En este capítulo haremos una introducción al marco teórico más amplio, sobre la simulación en general y en particular la problemática del dominio aeronáutico. Esta base teórica general constituye el marco en el que se encuadra este trabajo.

## 2.1 - Generalidades

Una forma de dar comienzo a este trabajo es tomar la definición de simulación del diccionario de la Real Academia Española: "Simular es presentar una cosa, fingiendo o imitando lo que no es". Por lo tanto, una definición de simulación puede consistir en el empleo de un modelo de sistema; cuyo interés se centra en un aspecto específico, real y observable; con el propósito de colaborar en un proceso de investigación, experimentación y/o educación. De la misma forma podemos referirnos al equipo o sistema mismo como el "simulador".

Conceptualmente como es conocida actualmente, nace en 1929 cuando el Ing. Edwin A. Link, logró poner en funcionamiento el primer simulador de vuelo. Es decir el primer simulador bajo la concepción actual, ha sido desarrollado en el dominio aeronáutico. Pero el impulso de la herramienta como la conocemos en nuestros días se sitúa en la Segunda Guerra Mundial, hasta generalizar el concepto actual que la liga a un computador y / o dispositivos electromecánicos.

Ubicar el origen de la simulación en la industria aeronáutica no es arbitrario, sino que es un dominio que, por sus características, justifica ampliamente el uso de la misma ya que, tanto en el ámbito Universitario como en la industria Aeronáutica es necesario contrastar los resultados teóricos con la práctica, pero lamentablemente, en los aspectos relacionados

con la aerodinámica aplicada y la estabilidad y control, la única posibilidad sería la de construir los prototipos de aeronaves en estudio.

En el campo del diseño, durante muchos años los Túneles Aerodinámicos, fueron la herramienta disponible más utilizada, sin embargo en la actualidad los desarrollos numéricos y la posibilidad de respuesta en tiempo real, ha permitido encarar desarrollos a partir del año 1991/1992, utilizando simuladores como verificación previa a la construcción de los prototipos lo cual, en muchos casos, constituye el único complemento posible a los tratamientos analíticos clásicos. Entonces, la **Simulación del comportamiento dinámico de un avión en la fase de desarrollo** tiene por objeto prever el comportamiento dinámico de aeronaves en desarrollo en la etapa avanzada, mediante el uso de potentes ordenadores que mediante programas adecuados pueden predecir el comportamiento en vuelo de la futura aeronave. Este tipo de simulación es de reciente utilización y tiene por sus características una inmejorable relación con aspectos académicos y de formación profesional, siendo de especial aplicación académica ya que permitiría simular integralmente las características de vuelo de los proyectos desarrollados por alumnos, como método de comparación de resultados de desarrollos analíticos, en general para aspectos particulares del diseño, y como respuesta integral del comportamiento de la aeronave.

Cabe citar como antecedente para la aplicación académica de este tipo de simuladores, al igual que en desarrollos de ingeniería, los trabajos presentados en el **II Congreso Nacional de la Ingeniería Aeronáutica**, celebrado en España (Madrid 15/19 Nov-93) presentados por los Ing. J. Lapastora Turpin, y J. Vigil de la Villa (C.A.S.A.). También vale citar como antecedente el simulador desarrollado con fines didácticos por el **Departamento de Aeronáutica y Astronáutica del Instituto Tecnológico de Massachussets (Massachussets Institute of Technology-MIT),** el cual responde más a aspectos relacionados con la aproximación por instrumentos y la generación de imágenes ambientales exteriores, con bastante menos precisión en los aspectos aerodinámicos; y el proyecto **LaRS++ (Langley Standard Real-Time Simulation in C++ framework – 1995 - NASA)**, el cual constituyó el paso del framework denominado LaRC, desde FORTRAN procedural, a Orientación a Objetos en C++ denominado LaRS++, y es utilizado principalmente para probar sistemas experimentales que han demostrado trasladarse a la aeronave sin necesidad de modificaciones gracias a la orientación a objetos.

Analizando los simuladores para diseños aeronáuticos se podrá notar que sólo organizaciones de gran envergadura poseen desarrollos serios de este tipo, los cuales evidentemente no se encuentran disponibles fuera de los mencionados propietarios.

Si incorporamos también a los **simuladores de vuelo**, la gran mayoría de desarrollos en este campo son simuladores ineficientes; simuladores semejantes a juegos; desarrollos complejos utilizando programación secuencial; simuladores basados en aeronaves existentes; etc. Además la mayor parte del software está realizado en lenguaje procedural requiriendo un alto esfuerzo el modificar y mantener dicho código, sin mencionar los problemas de eficiencia que esto incorpora. Estos últimos puntos se encontrarán como problemas centrales en toda la bibliografía sobre simulación ya que, una de las características inherente a este tipo de sistemas es el cambio constante de estructuras complejas propias del dominio de aplicación, lo cual hace mandatario el reuso, fácil mantenimiento y extensión.

A continuación abarcaremos una introducción teórica sobre los conceptos que forman el dominio de estudio, la cual ayudará a clarificar ciertos conocimientos sobre el dominio que estamos analizando.

## 2.2 - Importancia y Papel de la Simulación

Evidentemente de lo expuesto hasta ahora surge la importancia en costos y posibilidades que presenta el uso de la simulación en el ámbito aeronáutico.

Evidentemente existen restricciones y puntos a tener en cuenta. Partiendo de la base que la simulación es un método experimental, para experimentar con un modelo de simulación en lugar de utilizar el modelo real (cuyo diseño es además el punto clave en estudios de simulación). Los factores de riesgo son:

- La creación de los modelos es muy demandante.
- Conocimiento limitado del sistema a simular.
- Cómputos complejos que toman mucho tiempo (Tiempo real).

Por lo tanto existe una regla que postula: “Si el experimento es posible, realícelo. Es siempre el mejor método porque todos los aspectos son considerados. Aunque otros métodos se utilicen durante la etapa de diseño, el experimento puede servir como evaluación final del sistema. Si el experimento no es posible, trate de encontrar un método analítico. Si no hay disponible, use simulación”.

La simulación no es solo el último recurso como se postula en la regla anterior. La simulación puede contribuir a entender mejor el sistema no solo brindando respuestas a las preguntas originalmente planteadas, sino que generalmente la creación del modelo de simulación es la primera ocasión donde ciertas cosas son tomadas en cuenta.

Algunos beneficios es reducir gastos por el uso de munición real; disminuir el desgaste y deterioro de armas y sistemas electromecánicos; Las destrezas de las tripulaciones en el uso de los sistemas de la aeronave, en tareas individuales y colectivas. Aquí es donde encontramos uno de los puntos más fuertes: las aeronaves miden su vida en horas de vuelo, ciclos de aterrizaje y despegue entre otros parámetros. Avanzar en estos parámetros implica costos importantes de mantenimiento, seguridad, amortización que pueden ser reducidos por el uso de simuladores. No valorar la influencia que tiene esta herramienta en la formación de tripulaciones es camino seguro al deterioro progresivo del material como también a no explotar todas las capacidades que posee.

Ante la necesidad de modelar lo más cercanamente posible la realidad, la simulación ha penetrado en todo tipo de campos a un ritmo de avance similar a la tecnología más vanguardista. Así tenemos:

- Militar: Se utilizan mayormente en el área de capacitación/investigación y se pueden dividir en 3 categorías a fin seguir un patrón de aprendizaje progresivo:
    - De Función específica o didácticos: introducir a un alumno en el funcionamiento de un subsistema. Ejemplos son:
        1. Cargar un arma; Polígono de tiro.
        2. Entrenamiento de Instrumental de un vehículo.
        3. Reconocimiento y dominio de la cabina de un helicóptero.

4. Lecciones básicas para conducir un vehículo blindado.
5. Tiro reducido de morteros por diversos medios.

- De Entrenamiento táctico-técnico: Persiguen la comprobación de acciones en el marco de ejercicio de tropas permitiendo seleccionar, evaluar y capacitar al personal en posibles escenarios de contienda futuros. Estos implican la instalación de dispositivos de tiro y sensores en medios reales (BT46 sueco; SimFire inglés; Miles EEUU).
- De puestos de mando; control: Para formar comandantes y especialistas mediante el empleo de computadoras para recrear el ambiente de combate, como por ejemplo de guerra electrónica. (Army Training Battle Simulation System (ARTBASS); Battalion Automated Battle Simulation (BABAS); Corps Battle Simulation (CBS); Computarized Battle Simulation (COMBAT-SIM)).

- Civil: Es conocida la gran variedad de aplicaciones, pero para el tema que nos compete podemos mencionar:
  - simuladores de vuelo para el entrenamiento de pilotos
  - simuladores de mantenimiento para el personal técnico
  - simuladores de tráfico aéreo (ATC) para los controladores de vuelo.

## 2.3 - Tipos de Simuladores en Aeronáutica

Básicamente existen cuatro tipos de Simuladores, en orden de complejidad creciente, cumpliendo con las prestaciones de:

1. Reproducir en tamaño real la cabina de una aeronave en desarrollo, para poder ubicar los equipos, llaves, asientos, etc.: Este tipo de simuladores es conocido como ***Mockup*** y carecen de importancia para los objetivos de esta tesis, ya que son esencialmente destinados al desarrollo de un avión en particular. También se los usa para aeronaves específicas ya construidas, para instruir a las

tripulaciones en la ubicación de los elementos de la cabina (estática).

2. Simulación del Vuelo de un Avión General: Destinado fundamentalmente al entrenamiento de pilotos en maniobras de vuelo y navegación aérea. Los aspectos dinámicos, de performance y operación no responden específicamente a ningún avión en especial, las respuestas (reacciones del simulador) son genéricas y su aplicación es exclusiva para entrenamiento básico de pilotos en el Vuelo Instrumental, por ello se los llama **Entrenadores.**
3. Simulación del vuelo de un avión particular: Similar al del punto anterior, pero con mucha mayor complejidad, ya que representa a un avión en particular, los aspectos dinámicos y de performance están almacenados en tablas que reproducen aspectos del vuelo real de la aeronave que simula. Se los usa para el entrenamiento de pilotos en ese avión en particular. Se los conoce como **simuladores de vuelo**, y en general simulan sensaciones de fuerzas y aceleraciones (**Simuladores dinámicos**).
4. Simulación del comportamiento dinámico de un avión en la fase de desarrollo: Tienen por objeto prever el comportamiento dinámico de aeronaves en desarrollo en la etapa avanzada, están controlados por potentes ordenadores que mediante programas adecuados pueden predecir el comportamiento en vuelo de la futura aeronave. ***Son de reciente utilización y tienen por sus características una inmejorable relación con aspectos académicos y de formación profesional.***

De estos tipos de simuladores, el cuarto es el que reviste interés a los fines de esta tesis.

## 2.4 - Tipos de Simulación

### 2.4.1 -Simulación Continua

La simulación de tipo continua, nace con los lenguajes de simulación desarrollados al final de los 50 bajo la forma de simuladores desarrollados con computadoras analógicas. Se basa en crear un sistema electrónico analógico

cuyo comportamiento se describe por el mismo modelo matemático (ecuaciones diferenciales) que el sistema siendo investigado. El sistema electrónico se crea interconectando bloques estándar basados en amplificadores operacionales modificados para actuar como integradores, sumadores, y otras unidades funcionales. Entonces el usuario realiza experimentos con esos sistemas electrónicos aplicando entradas y midiendo el voltaje en ciertos puntos de salida (osciloscopio, plotter). El voltaje cambiante representa una función del tiempo, que es la misma que describe los cambios en el sistema original cuya naturaleza física puede ser totalmente diferente (desplazamiento mecánico, temperatura, etc.). Las computadoras analógicas realizan la función de integración mejor que las digitales que la realizan de forma más lenta y menos precisa. Algunas aplicaciones especiales se basan en el uso de computadores híbridos, que contienen partes analógicas y digitales conectadas por conversores A/D y D/A. La parte digital hace todo menos integrar.

### 2.4.2 -Simulación Discreta

Los sistemas cuya dinámica se puede considerar como una secuencia discreta de eventos ocurriendo en puntos discretos de tiempo, responderán a un modelo de simulación de tipo discreta. La clave de los lenguajes de simulación discreta es la forma en que controlan la secuencia apropiada de actividades en el modelo.

### 2.4.3 -Simulación Orientada a Objetos

La simulación del tipo Orientada a Objetos, puede ser considerada como una clase especial de OOP. Algunos principios de OOP como la existencia de un numero variable de instancias de objetos han sido un estándar utilizado en simulación hace mas de 30 años (Simula 67). Algunos conceptos como clases, herencia, métodos virtuales, etc., han sido definidos en Simula mucho tiempo antes que sean redescubiertos en OOP.

Algunas características comúnmente aceptadas en simulación Orientada a objetos son:

- el algoritmo o dinámica del sistema se expresa en términos de objetos (actores) que existen en paralelo y que interactúan uno con otros. Cada objeto se representa por: parámetros, atributos, métodos, vida, que representa la actividad iniciada desde la creación del objeto.

- Los objetos pueden interactuar de esta forma:
    - acceso directo a parámetros y atributos.
    - Llamada mutua a métodos
    - Comunicación y sincronización de las vidas de los objetos.
- Los objetos similares son agrupados en clases también llamadas prototipos.
- Los objetos pueden ser clasificados jerárquicamente por la llamada herencia o subclases.
- La existencia paralela de objetos necesita estructuras que soporten la cooperación y sincronización de sus vidas. La vida de un objeto no necesariamente tiene una dimensión de temporal, pero en el caso de OOS la tiene.

### 2.4.4 -Simulación en línea

Básicamente surgida con el uso de Internet junto a Java y JavaScript, siendo utilizada para determinados tipos de problemas. En vez de descargar e instalar paquetes de software, es posible abrir directamente calculadoras, especialmente en problemas poco frecuentes o que no requieran cómputos complejos y ejecutar los cálculos directamente utilizando el poder de cálculo de la PC local o bien del servidor. En general no es aplicada a los problemas de tiempo real como la simulación de vuelo.

## 2.5 - Tratamiento del Tiempo

Como se debe reflejar el mundo se deben crear comportamientos del mundo real basados en tiempo. O sea al activar un control la respuesta debe ser la misma y en el mismo tiempo que la realidad. “Mismo tiempo” implica dentro de un margen superior e inferior de tiempo (responder muy rápido es igual de malo que responder muy lento para la calidad de la simulación).

Hay dos formas de manejar el tiempo: Periódico (Periodic) cuando se debe mantener el tiempo real (aeronave) y Basado en eventos (Event-based) cuando el tiempo real (real-time) no es critico (estación de instructor).

### 2.5.1 -Periódico

El tratamiento periódico del tiempo consiste básicamente en que la ejecución se encuentra dividida en una cantidad fija de tramas de tiempo (frame-rate). Este esquema usa una disciplina cíclica de scheduling que procede de la iteración, a través del siguiente ciclo:

- Fijar el tiempo inicial simulado
- Iterar los siguientes 2 pasos hasta que la sesión se complete
- Invocar cada proceso por un quantum fijo (real). Cada proceso calcula su estado basado en el tiempo actual simulado y reporta basado en el periodo de tiempo simulado siguiente. El garantiza completar su computación dentro del real-time quantum.
- Incrementar el tiempo simulado un quantum.

Mientras cada proceso lo pueda lograr en el quantum asignado podrá continuar la simulación y es responsabilidad del diseñador proveer los procesadores necesarios para que los módulos (pequeños como sean necesarios) logren sus cómputos en el quantum asignado.

### 2.5.2 - Basado en Eventos

El tratamiento del tiempo basado en eventos procesa la ocurrencia de eventos e itera en el siguiente ciclo:

- Agregar un evento simulado a la cola de eventos
- Mientras haya eventos en la cola
- Elegir el evento con el menor tiempo simulado (el mas cercano)
- Fijar el tiempo simulado al tiempo de ese evento
- Invocar un proceso para el evento elegido.

La simulación avanza a medida que se ubican eventos en la cola. Estas pueden correr más rápido o mas lento que el tiempo real.

### 2.5.3 - Mixto

Los simuladores deben acoplar regímenes distintos como son los eventos del instructor (eventos) y los del modelo de la aeronave (periódico).

Un método simple es que el procesamiento periódico ocurre inmediatamente luego de un paso de sincronización y se completa antes de

un procesamiento no periódico. El procesamiento no periódico procede dentro de un intervalo durante el cual se deben procesar tantos mensajes como se pueda y los restantes serán procesados en los intervalos subsiguientes.

Los estados de la simulación se calculan por períodos discretos los cuales deben ser lo suficientemente cortos para lograr el realismo necesario. O sea, existe una dependencia entre el realismo deseado o necesario y los períodos de tiempo involucrados en el cálculo de la simulación:

- Los cálculos para cada componente se debe realizar en ese período.
- Un componente puede afectar otro por lo cual deben transmitirse sus estados.
- En vista que los sistemas del avión trabajan en un tiempo dado, ese tiempo debe ser simulado.

El procesamiento se realiza de la siguiente manera:

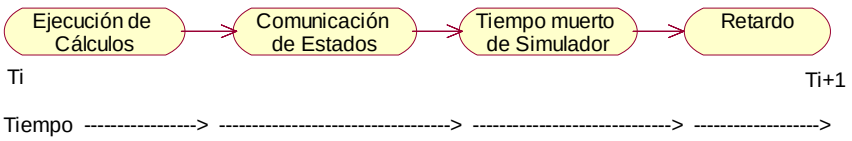


**Figura 1 Ventana de Tiempo**

Aquí se ve el denominado “data coherent problem”, dónde se requiere que la información este presente al comienzo de cada período, sino los cálculos serán erróneos.

## 2.6 - Características de las Aeronaves

El objeto sobre el cual se aplicará todo el desarrollo de la tesis (desde la óptica informática), son las aeronaves. En el sentido más amplio de “aeronaves”, se pueden incluir todo vehículo espacial, incluyendo aquellos que operan en la exosfera, o interactúan con la atmósfera terrestre, como los transbordadores espaciales, misiles o vehículos no tripulados de cualquier tipo

y con cualquier característica, siendo las leyes adaptables a cualquier atmósfera hipotética, tal como la marciana.

Reuniendo a todos estos vehículos bajo el concepto genérico de aeronaves, es conveniente analizar sus características, ya que son las que fijan las condiciones de contorno que enmarcan el trabajo de Tesis y el enorme grado de complejidad asociado.

Las Aeronaves son vehículos que a diferencia de muchos otros, solo son comparables en su concepción filosófica a los submarinos, esto es se mueven en tres dimensiones, lo que genera desde la óptica de la Mecánica Física, un tratamiento de la dinámica mucho más complejo. Un móvil que se mueva en el espacio de tres dimensiones, puede tratársela desde la óptica teórica de la Mecánica Newtoniana, o Lagrangeniana, con sistemas de ecuaciones diferenciales. Si el cuerpo es rígido, y admitimos tres grados de libertad adicionales, correspondientes a otros tantos controles cartesianos cabeceo, rolido, y guiñada (pitch, roll y yaw), constituyen un sistema de 9 ecuaciones diferenciales simultáneas, que requieren de una resolución matricial.

Estas matrices, por razones prácticas y de posibilidad de ser utilizadas para resolver problemas numéricos, deben ser cuadradas, por lo que los modelos matemáticos de los fenómenos dinámicos, se ven reducidos a solo nueve variables independientes, (nueve ecuaciones), aunque se pueda utilizar adicionalmente las derivadas temporales primera (velocidad) y segunda (aceleración) de éstas.

Sin embargo, sigue existiendo una insalvable limitación: Las ecuaciones diferenciales deben ser lineales, esto es, los coeficientes que caracterizan a cada ecuación diferencial, no deben ser función de las variables de estudio. Esta limitación, para las herramientas analíticas disponibles, es algo prácticamente insalvable, exceptuados casos muy particulares cuya solución es conocida como método de Lyapunov, muy complejo de aplicar en sistemas de ecuaciones.

La dinámica real de una Aeronave es muchísimo mas compleja que estas posibilidades: de hecho podríamos identificar mas de cien variables independientes que afectan al movimiento de una aeronave en el espacio, en especial si incluimos, variación de los parámetros termodinámicos de la atmósfera (p,T,R, y densidad ($\rho$)), de los parámetros físicos de la misma (Vientos, ráfagas, corrientes de chorro, térmicas, corrientes convectivas, etc), de los cambios de estado de la atmósfera, lluvia hielo, viscosidad,

características mecánicas del planeta, aceleración de Coriolis, etc. Aunque estos fenómenos pudiesen ser modelizados y resueltos, no se podrían introducir en un sistema de nueve ecuaciones, por cuanto, como se ha dicho, las variables deben ser solo nueve. Además su resolución impide las modificaciones aleatorias de cualquiera de ellos, es decir NO EXISTE LA POSIBILIDAD DE INTERACCIÓN entre el hombre y el sistema, indispensables en un simulador interactivo.

Sin embargo a fin de generar una conciencia adecuada acerca de la complejidad del sistema avión, podemos recurrir a una síntesis de los fenómenos a los que nos enfrentamos:

a) Las aeronaves están sometidas a la interacción derivada de su movimiento, con la atmósfera que lo rodea. Esta interacción genera fuerzas, y estas generan momentos, los que a su vez generan cambios en las actitudes, y velocidades de estos cambios, que finalmente generan cambios en las fuerzas y momentos y así sucesivamente.

b) En su movimiento dentro de la atmósfera, la aeronave recorre distintos puntos de ella en los que sus características termodinámicas varían respondiendo a la ecuación de Estado en todo punto, pero, la variación de estos parámetros individualmente entre puntos contiguos, no obedece a leyes simples, y aunque se presentan como aleatorias, podrían analizarse desde la estadística o desde la Teoría del Caos, en forma mas avanzada. Esto nuevamente atenta contra la posibilidad de interacción.

c) El movimiento de la aeronave está sujeto a las leyes de la Mecánica, esto es, por tener una masa, tiene Inercia y por ello suceden fenómenos de interacción energética entre Potencial y Cinética.

d) Por ser dichos cambios previsibles respecto de la terna que rodea a la aeronave, que no es Inercial, hay que referirlos a una terna que podamos considerar inercial, p.e., la Tierra, lo que nos obliga a manejar todas las ecuaciones de arrastre, Coriolis, etc.

e) Todo lo anterior surge de Considerar a la aeronave como rígida, lo que en general es irreal. Para afinar la trama necesitamos incluir

fenómenos elásticos estructurales (aeroelasticidad), y fenómenos derivados de la Aerodinámica No Estacionaria.

f) Los coeficientes aerodinámicos se mantienen lineales con determinados parámetros de estudio, por muy poco tiempo. La alinealidad se hace muy notoria, y genera complejos fenómenos de control que han producido muchísimos accidentes catastróficos.

g) En su movimiento relativo con la atmósfera, la Aeronave varía su velocidad. En los rangos subsónicos bajos (máximo 400 Km/h), podemos considerar que las características termodinámicas de la atmósfera que rodea a la aeronave, no cambian con la velocidad, y en los distintos puntos de la aeronave posee plena vigencia el Nro. de Reynolds, las leyes de Similitud, etc., sin embargo si las velocidades superan este valor, los parámetros termodinámicos que rodean al avión ya no son constantes, y varían según otras leyes, surgiendo correcciones de Prandtl, Glauert, Meyer, Von Karman, Tsien, etc., entrando en vigencia otros parámetros relevantes como el Nro de Mach, derivados todos ellos de los fenómenos locales de compresibilidad del aire, donde su densidad deja de ser constante.

h) Si las velocidades relativas siguen aumentando surgen discontinuidades. Las más notorias son las Ondas de Choque. Las leyes cambian abruptamente, y los coeficientes aerodinámicos, CL, CD, Y Cm, con ellos, entrando en vigencia las relaciones de Mach, que intentan describir matemáticamente estos cambios, que generan nuevas alinealidades que complican su tratamiento (Pérdidas de Alta y Baja Velocidad, buffeting, flutter etc.).

i) Si prosigue el incremento de la velocidad, a velocidades cercanas a Mach 5, el flujo sufre otro cambio físico y conceptual, el aire deja de comportarse como un continuo y comienzan fenómenos de ionización, altísimas temperaturas locales y cambios e la densidad y la presión.

j) La interacción del piloto, a través de los controles, produce cambios aleatorios que son objeto de Estudio en los Centros de Ensayos en Vuelo.

Todas estas variables son imposibles de introducir en un sistema rígido de 9 ecuaciones diferenciales simultáneas. Además las soluciones matemáticas analíticas, entregan una respuesta para todo “t”, y no es posible

introducir en el “lazo” del cálculo, la voluntad aleatoria de cambiar alguna constante del sistema.

## 2.7 - Solución Tradicional

Definitivamente, la solución tradicional de integrar ecuaciones diferenciales acopladas, mediante el análisis matricial y obtención de la “Ecuación característica“ del sistema, cuyas raíces, en general producen Pares Complejos Conjugados, y raíces reales, y representan las armónicas del movimiento que sumadas dan como resultado las respuestas temporales de las variables de estudio, están acotadas por una serie de “redondeos” adicionales, que son los que se utilizan permanentemente en la Ingeniería, estos son, considerar ángulos pequeños, que esencialmente significan que:

Sen $\alpha = \alpha$

Cos $\alpha = 1$

Estas limitaciones, impuestas por la imposibilidad de resolverlas de otra forma, sumadas a otras, como que las armónicas son las mismas para todas las variables de estudio, dan como resultado que este tratamiento solo permite predecir el comportamiento de aeronaves en un rango de performances muy acotado a condiciones de vuelo también acotadas, impidiendo el tratamiento donde los coeficientes de fuerzas y momentos dejan de ser lineales con los ángulos que los generan.

Sin embargo los fenómenos más interesantes de estudiar en una aeronave en desarrollo, y más peligrosos para la integridad física de la aeronave y personas, están en esta gama alineal, por lo que en los desarrollos se recurre a la experiencia en aviones similares y posteriores ensayos en vuelo.

Sin embargo este análisis cuyos primeros trabajos corresponden a Perkins, Hage, Etkin, Glauert y otros, signaron los desarrollos aeronáuticos hasta 1990, aún en aviones como Boeing 737, 727, 707, 747, Douglas DC-10, MD-11, DC-9, aeronaves como los Antonov 124 y 225 (Los mas grandes del mundo, con cerca de 750 toneladas al despegue), o aeronaves de muy alta

velocidad, como el Concorde recientemente retirado, o de combate de todo tipo, etc.

Sin embargo todos los procesos de diseño implicaban construir prototipos y probar todas aquellas características imposibles de predecir por métodos analíticos, debidos a las limitaciones de los modelos matemáticos, ya explicados.

La variables de estudio tradicionales se suscribieron (y aún se utilizan en muchos casos) a las siguientes:

| | |
|---|---|
| $\alpha$ | Ángulos Aerodinámico de ataque. |
| $\beta$ | Ángulo Aerodinámico de deslizamiento |
| $\psi$ | Ángulo Aerodinámico de guiñada. |
| $\phi$ | Ángulo de inclinación lateral del avión. |
| $\theta$ | Angulo de cabeceo del avión. |
| **V** | Velocidad de Avance del Avión. |
| $\delta$**a** | Deflexión de alerones |
| $\delta$**e** | Deflexión de elevador |
| $\delta$**r** | Deflexión de timón de dirección |

Las ecuaciones de la mecánica forman una matriz, donde el resto de las variables se establecen como constantes para, estimando sus valores para ¨t¨ aproximándose a infinito [A26].

La solución del determinante de esta matriz, da como resultado la denominada **ecuación característica,** que define la cantidad y características de las “armónicas” que definen la dinámica del sistema llamado aeronave.

En particular esta Ecuación Característica, define la frecuencia y el amortiguamiento de éstas. El paso siguiente es la construcción del prototipo y volarlo.

Si bien este procedimiento ha sido mas o menos constante desde la primera guerra mundial hasta los ’90, aún así, estudiar determinadas característica del vuelo, cercanas a los condiciones límites de seguridad, entrañaban un elevado costo y no pocas vidas humanas.

## 2.8 - Introducción a la Física del Problema

Anteriormente se ha mencionado la importancia de los túneles aerodinámicos, ya que permiten el **estudio de fenómenos dinámicos** (estabilidad y control), de modelos completos a escala para obtener respuestas temporales de un grupo de variables de estudio, para condiciones de vuelo simétricas, **y ciertas mediciones estáticas, como la Resistencia al Avance y Sustentación** y Momentos para toda condición de vuelo (modelos completos).

El objeto de estas mediciones en túnel está asociado al hecho que como la gran mayoría de los efectos aerodinámicos son muy difíciles de determinar con precisión, tales como la interferencia ala fuselaje, down wash (deformación de la corriente detrás del ala), etc.

Estas dificultades para determinar fenómenos aerodinámicos se extienden a la determinación con precisión los coeficientes que caracterizan al sistema dinámico (en general, Derivadas Parciales) en su estudio clásico de ecuaciones diferenciales.

Uno de los principales inconvenientes técnicos a resolver, asociados a este intento, corresponde al hecho que los fenómenos a medir están relacionados con ciertos números adimensionales tales como el número de Reynolds (Re), o el número de Mach (M), dependiendo su influencia de las velocidades relativas entre la vena fluida, y el modelo en estudio.

$$M = V / C$$

Y

$$Re = V L \rho / \mu$$

Estas dos fórmulas, sintetizan el problema planteado; es decir, que cuando se estudia un modelo a escala, para que los resultados obtenidos (en el campo de las fuerzas aerodinámicas), sean coherentes con el avión que el modelo pretende representar, debe verificarse la **constancia del número de Reynolds**, lo que implica que, a menor escala del modelo, es necesaria una mayor velocidad relativa de la vena fluida dentro del túnel (ya que sería impráctico mantener dicha constancia utilizando un fluido distinto al aire, distinto $\rho / \mu$, dentro del túnel, por infinitas razones).

Esto para bajas velocidades no representa un problema de envergadura, pero sin embargo hay que tener en cuenta que para que un avión que en la realidad vuela a 200 km/h, y el modelo sea a una escala lineal de 1/10, la velocidad de la vena dentro del túnel deberá ser 10 veces superior (2000 km/h), para poder mantener constante el numero de ***Re.*** lo que hace entrar en escena a los problemas asociados **al número de Mach,** obviamente aún muchísimo antes de esta velocidad, es decir problemas de ***compresibilidad.***

Si a esto le asociamos el hecho de la necesidad de una cámara suficientemente alta para permitir el movimiento vertical del modelo, sin alcanzar la zona del efecto suelo (o techo) de la cámara, nos da como resultado la necesidad de una gran sección de la cámara, perpendicular a la vena de aire. Este problema representa en síntesis la necesidad de una gran potencia instalada, para mover la vena de aire requerida dentro del túnel, y la enorme dificultad técnica y económica .

El planteo de los métodos numéricos en general, convergen a sistemas matriciales, cuyo orden en general esta relacionado con la complejidad del problema, tal como ocurre en los estudios dinámicos de los aviones, en los que conviven fenómenos de distinta índole en forma acoplada y teniendo en cuenta que **deben ser en tiempo real**, se genera el problema que los planteos matriciales en general requieren de un tiempo de procesamiento proporcional al orden de las matrices en cuestión. Estas soluciones requieren hardware muy potente no disponible generalmente.

Una solución planteada por el Grupo de Simulación Dinámica del Vuelo ha optado por un ***un modelo similar al sistema cinematográfico, en el que los cambios de los parámetros de estudio, en el tiempo, se analizan cuadro a cuadro.***

De todas formas para lograr la modelización completa de los fenómenos dinámicos asociados a un avión, se hace necesaria la realización de modelizaciones numéricas previas de datos contenidos en Reports. Entre estos desarrollos el grupo ha desarrollado: **modelización numérica de la contribución del fuselaje a la estabilidad longitudinal, por efectos del up, y down wash del ala**, sobre la hipótesis del NACA TM 1036**. Determinación Numérica Tridimensional del valor del Down Wash**, a partir del planteo de Biot-Savart, **Generalización del modelo matricial de Glauert *para Distribución de Sustentación Asimétrica***, **originada en asimetrías aerodinámicas, geométricas, y/o funcionales.**

## 2.9 - Problemas y Limitaciones

Las respuestas dinámicas de cualquier sistema con masa no nula, puede ser representadas por funciones armónicas superpuestas. Estas son en general funciones senoidales, cuya amplitud y amortiguamiento varían con las características de las aeronaves.

Las respuestas temporales reales de las variables de estudio antes mencionadas (y de aquellas que no se pueden determinar por la problemática antes mencionada de los modelos matriciales), en general poseen frecuencias y amortiguamiento acotados, es decir podemos afirmar que los períodos de las armónicas van desde el **medio segundo,** hasta **infinito.** Obviamente la complejidad reside en determinar la forma de las armónicas que corresponden a los períodos más cortos.

La solución de las ecuaciones diferenciales, permite trabajar sobre estas formas con un grado de certeza que reside en las constantes que caracterizan a las mismas: las soluciones serán más aproximadas y alejadas de la realidad, si no pueden aceptar valores de coeficientes variables.

Existen muchos métodos numéricos que permiten introducir cambios en las constantes del sistema, resolviendo los mismos en forma iterativa, afinado la trama que modela el fenómeno, pero, estos métodos requieren mucho tiempo para el cálculo, por lo que en general, calcular un segundo de simulación del fenómeno podría tomar media hora, lo que elimina la posibilidad de interacción en tiempo real con el modelo.

## 2.10 - Conclusiones Preliminares

El desarrollo de simuladores de aeronaves en la etapa de diseño nos presenta cierta cantidad de desafíos:

a) **Trabajar en tiempo Real**. Esto es, los fenómenos calculados para un segundo, deben tomar menos de ese tiempo para que pueda haber interacción con la simulación (caso de querer volar e interactuar con la aeronave en estudio).

b) **Calcular suficientes cantidades de puntos para un función determinada.** Se considera que un ciclo senoidal está suficientemente definido si conocemos diez puntos por cada semiciclo, o sea que si

como hemos dicho éste tiene un período de 0.5 segundos, necesitaremos 20 puntos equiespaciados cada 2.5 centésimas de segundos. Con el fin de ser conservativos, hemos considerado como máximo un recálculo de todas las **variables cada 0,017 segundos (60 HTZ).**

c) El Grupo de Simulación dinámica del vuelo ha optado por un sistema de secuencia abierta (no matricial), con integración secuencial (efecto cinematográfico). Esta integración secuencial, posee como principio filosófico el de establecer un camino de renovación de datos y al cabo del mismo, se ha pasado por todas las variables una sola vez, y se repite la secuencia. La definición anterior implica que todas las variables se calculan utilizando parte de datos "viejos" y parte de datos "nuevos", emulando el principio televisivo del **"entrelazado".**

d) Se considera válido el método cuando: Las respuestas obtenidas no dependan de la secuencia de integración; El simulador responda a las características de una aeronave conocida; Las respuestas temporales obtenidas no cambien al reducir el frame de tiempo ($\Delta t$).

# Capítulo 3 Patrones de Diseño

*"OO methodologies involve a lot more than programming techniques"* [YOU94]

Continuando con la presentación de los conceptos teóricos necesarios para el desarrollo de la presente tesis, en este capítulo haremos una introducción sobre Patrones de diseño. Se trata solo de una breve reseña para hacer presente los conceptos básicos que son necesarios aquí.

## 3.1 - Introducción

Comenzaremos definiendo Patrones de Diseños como una solución abstracta a un problema recurrente en un dominio específico. Los patrones de diseño están ganando mayor aceptación en la comunidad de software y se están convirtiendo en un importante elemento en la construcción de software moderno. Cabe remarcar que capturan prácticas existentes documentando presunciones, estructura, dinámica y consecuencias de una decisión de diseño. El principal objetivo de un patrón de diseño es comunicar los detalles de un diseño.

No hay que limitar el uso de patrones al paradigma de objetos ya que han tenido buenos resultados en otros contextos. Como se verá a continuación la historia de los patrones de diseño comienza fuera de la orientación a objetos.

## 3.2 - Historia

Se originan en el trabajo del arquitecto Christopher Alexander, quien a finales de los 1970's publicó dos libros sobre patrones de diseño para planeamiento urbano y arquitectura de edificios, A Pattern Language y A Tímeles Way of Building. Alexander buscaba crear estructuras que fueran buenas para la gente mejorando su confort y la calidad de vida.

Básicamente, Alexander se planteaba qué elementos preguntaba están presentes en un diseño de buena calidad que le faltaba a los diseños mediocres. Alexander estudió esto haciendo observaciones de edificios, ciudades, calles y otros aspectos de los lugares que el hombre construyó para vivir. Al final descubrió que las buenas obras arquitectónicas tenían factores en común. Esas similitudes las llamó Patterns (patrones) y los definió como una solución a un problema en un contexto: Cada pattern describe un problema que ocurre una y otra vez en nuestro ambiente, y entonces definió una solución a ese problema, de una forma en que pueda ser utilizada nuevamente una y otra vez sin repetir la forma de realizarla.

Los cuatro elementos requeridos para describir un pattern según Alexander son:

- Nombre.
- Propósito, problema que resuelve.
- Cómo logra su propósito.
- Restricciones y fuerzas a considerar.

En el comienzo de los 90s, algunos desarrolladores experimentados de software tomaron el trabajo de Alexander y se preguntaron si también podrían aplicar este concepto a los problemas de diseño de software que ocurren una y otra vez. Han concluido que esto es así y muchos han trabajado desde entonces, pero fue en 1995 cuando se publicó el libro “Design Patterns: Elements of Reusable Object-Oriented Software” por “the Gang of Tour” el cual es considerado el libro más popular en ese campo con gran influencia en la comunidad.

## 3.3 - Definición

El patrón de diseño es la abstracción de una forma concreta que se mantiene recurrente en contextos no arbitrarios específicos.

No son solo soluciones probadas para un problema recurrente, sino que el problema ocurre dentro de un contexto, y ante la presencia de varios

factores de competencia. Esta solución involucra alguna forma de estructura que es un balance de esos factores de la mejor forma para ese contexto.

Los patrones son dispositivos que permiten compartir conocimiento acerca del diseño de los programas. Documentando patrones se logra la forma de reutilizar y compartir información que se ha aprendido sobre la mejor forma de resolver un problema de diseño.

Para una mejor comprensión veamos que un patrón NO ES:

- ***Un algoritmo*** ya que resuelven problemas más finos como ordenar, buscar y no se involucran con temas como mantenimiento, adaptación, reuso de un diseño.
- ***Un framework*** ya que pueden utilizarse tanto en el diseño como la documentación del framework y este puede contener muchos patrones de diseño. Un framework puede verse como la implementación de un sistema de patrones de diseño.
- ***Un idioma*** ya que éstos son de menor nivel y más específicos del lenguaje de programación. (por ejemplo para administración de memoria en C++).

Otras definiciones incluyen:

- "Solucione recurrentes a problemas de diseño",et. al., 1998.
- "Conjunto de reglas describiendo como lograr ciertas tareas en el desarrollo de software", Pree, 1994.
- "Se centran en el reuso en temas de diseño arquitectónico recurrentes, mientras que los frameworks se centran en el diseño detallado y la implementación.", Coplien & Schmidt, 1995.
- "Resuelve un problema de diseño recurrente que se presenta en situaciones específicas y presenta una solución al mismo", Buschmann, et. al., 1996.
- "identifican y especifican abstracciones que están sobre el nivel de clases e instancias, o componentes.", Gamma, 1993.

Los patrones de diseño no tratan solamente sobre el diseño de objetos, sino de la comunicación entre ellos. En realidad son referidos como *communication patterns*, los cuales son importantes porque representan el diseño de simples, pero elegantes métodos de comunicación.

Los patrones de diseño pueden existir en varios niveles desde soluciones específicas a temas generales de sistema.

La mayoría de los cientos de patrones existentes fueron "descubiertos" en vez de ser simplemente escritos. El proceso de buscar estos patrones se llama "pattern mining" y es sujeto de un libro sobre el tema.

Los 23 patrones originalmente seleccionados fueron los que se conocían muchas aplicaciones conocidas y eran de un nivel medio de generalidad, donde podían abarcar varias áreas de aplicación y acompañar varios proyectos. Los autores los dividieron en:

- Creacionales: son aquellos que crean objetos, en vez de que sean instanciados directamente. Esto da al programa mayor flexibilidad en decidir que objetos necesitan ser creados para un caso dado.
- Estructurales: Ayudan a componer grupos de objetos en estructuras más grandes.
- Comportamentales: Ayudan a definir la comunicación entre objetos en su sistema y como el flujo es controlado en un programa complejo.

## 3.4 - Elementos

La forma más común para describir patrones ha sido presentada por Gang of Tour como:

| Campo | Descripción |
|---|---|
| Nombre | Debe poseer un nombre significativo para identificarlo. |
| Intención | Describe qué hace y qué problema de diseño resuelve. |

| | |
|---|---|
| Conocido como | Otros nombres conocidos en la literatura. |
| Motivación | Escenario que ilustra un problema y como el patrón lo resuelve |
| Aplicabilidad | Situaciones en que se utiliza. Generalmente son ejemplos |
| Estructura | Representación gráfica de los objetos en el patrón y sus relaciones |
| Participantes | objetos y sus responsabilidades |
| Colaboraciones | Como trabajan juntos los participantes para llevar a cabo sus responsabilidades |
| Consecuencias | Describe los resultados positivos y negativos de usar el patrón |
| Implementación | Consideraciones al momento de implementarlo: técnicas, trucos, etc. |
| Código de Ejemplo | Código que ilustra como se puede implementar el patrón de diseño. |
| Usos Conocidos | Describe ocurrencias conocidas y su aplicación en sistemas conocidos. |
| Patrones Relacionados | Relación entre este y otros patrones. |

## 3.5 - Razones para utilizar Patrones

Es la misma razón por la cual reutilizar buen código: beneficiarse del conocimiento y la experiencia de otra gente que se ha esforzado en comprender contextos, fuerzas y soluciones. Además, los patrones pueden ser más reutilizables que el código, ya que pueden ser adaptados para circunstancias particulares que no permiten reusar un componente existente.

- ***Mejorar la comunicación entre desarrolladores*** típicamente cuando se discuten soluciones alternativas a un problema.
- ***Han sido extraídos de diseños que se encuentran funcionando*** ya que capturan la esencia de un diseño para ser reutilizado en el futuro.
- ***Refuerza el reuso*** lo que ayuda a los desarrolladores inexpertos para lograr buenos diseños más rápidamente y constituyen un manual muy útil para la enseñanza de ingeniería de software. De todas formas no

son reglas a seguir ciegamente, sino que son una guía para alternativas probadas de diseño.

- ***Dan una perspectiva de alto nivel al problema*** ya que eliminan el problema de enfrentarse a los detalles tempranamente.

## 3.6 - Problemas

Los patrones no poseen una definición general aceptada y no existe estandarización, lo que implica que patrones escritos por uno u otro autor difieren en gran medida.

No existe una organización central que concentre y revise los patrones o los publique. Por eso se los encuentra en Internet con versiones diferentes llevando a la confusión.

Adoptarlos en una empresa es complejo ya que hay que generalizar su conocimiento y conocer gran cantidad para sacar realmente provecho.

Un programador experto que ha implementado una clase de software por años, no va a obtener beneficios extras, ya que ha escrito soluciones a ese problema con mayor rapidez y eficiencia sin el uso de ellos.

# Capítulo 4 Simuladores: Modelos Estructurales

*"A system build with OO methods is one whose components are encapsulated, which can inherit attributes and behavior from other such components, and whose components comunicate via messages with one another".* [YOU94]

En este capítulo finalizaremos la revisión teórica presentando los modelos estructurales utilizados históricamente en simuladores de aeronaves. En particular el AVSM lo veremos directamente con diagramas de clases adaptándolo al concepto de estructura de clases para que nos sirva como introducción al desarrollo que comenzará en el capítulo siguiente.

## 4.1 - Generalidades

El modelo estructural es un mapa arquitectónico para un sistema complejo de software o dominio [CRISPEN01]. El modelo que se ha generalizado y es utilizado actualmente en los simuladores, emergió del Ada Simulator Validation Program (ASVP – Dayton, OH, 1988), que estableció la eficacia de Ada para simulación de tiempo real de entrenamiento. Desde esa fecha se han realizado trabajos importantes como Structural Modelling del SEI y Mod Sim, entre otros. De esta forma el modelo estructural de los simuladores actuales emerge en la base de Ada como estándar de implementación y por lo tanto se ve influenciado en cierta manera por este.

El modelo estructural especifica las clases de entidades presentes en el diseño; como se representa el mundo en entidades de software y mecanismos de comunicación entre entidades. Es un estilo arquitectónico que soporta el cambio mediante la restricción de la estructura, comportamiento, y organización de los componentes de la arquitectura de software. El proceso involucra la partición del sistema y la restricción de la coordinación y comunicación entre las partes. Es una colección de tipos estructurales, unidos con las reglas de organización de las instancias de esos tipos en una arquitectura. Para resumir, trata de que los diseñadores y programadores mantengan sus diseños dentro de un grupo limitado de primitivas estructurales y de coordinación. Por lo tanto los principios se pueden definir como [KAZ01]:

- Partición predefinida de funcionalidad entre elementos de software.
- Restricción de datos y flujo de control
- Restricción a un número de elementos base
- Los componentes encapsulan su propio estado.
- Remoción de efectos laterales.
- Elimina la necesidad de diferentes niveles de empaquetado

En general basados en el concepto de Structural Modelling, han surgido dos modelos estructurales para simuladores: AVSM (Air Vehicle Structural Model), desarrollado por SEI en 1993 y DARTS (Domain Architecture for Reuse in Training Systems), desarrollado por Boeing.

## 4.2 - Arquitectura Basada en Flujo de Datos

Comenzando en 1960 con la construcción del UDOFT (Universal Digital Operacional Flight Trainer), el primer computador construido especialmente para el uso en simulación de aeronaves por Silvana Corporation, y luego con la llegada de computadores de uso general, se abre la era de la simulación digital. En esta primera etapa que podríamos situar en las décadas de los 60 y 70, el modelo estructural adoptado era una arquitectura basada en flujo de datos. Básicamente, esta elección concuerda, como se podría esperar, con los paradigmas de diseño y programación de la época.

En la siguiente figura se encuentran representados los elementos que intervenían en dicha estructura:

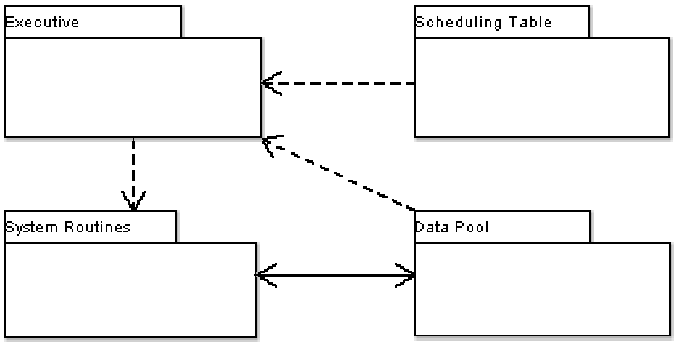


**Figura 2 Arquitectura Basada en Flujo de Datos**

Dónde:

- Executive: realiza la asignación de tiempo de las rutinas basado en la tabla de asignación.
- System Rutines: calculan el estado del simulador. En un principio principalmente, la resolución de las ecuaciones de movimiento de la aeronave y luego fue incorporando mayor funcionalidad.
- Scheduling table: requerimientos de asignación de tiempo de las rutinas del sistema.
- Data pool: área de datos que comparten las rutinas para intercambio de estado, etc.

Las rutinas del sistema representaban tareas completas de cálculo de estado como por ejemplo aterrizar. Si analizamos esta situación, intervienen ecuaciones y consideraciones propias de este aspecto del vuelo que estaban modelados en funciones que lo representaban.

En este modelo, no existía una estructura de la aeronave definida, ya que por ejemplo, si continuamos considerando el modelo que representa el aterrizaje, uno de los componentes que interviene es el tren de aterrizaje y por lo tanto el cálculo de su estado. Pero también tenemos la participación del tren de aterrizaje durante los despegues, lo que resultaba en una repetición de los cálculos de tren de aterrizaje tanto en la rutina de despegue como en la de aterrizaje.

Con el paso de los años y la evolución tecnológica, tanto desde el punto de vista de la Ingeniería de Software como también de la Ingeniería Aeronáutica, se van produciendo cambios evolutivos que se constituyen como

tendencia y rigen el estudio de este campo, los cuales pueden resumirse como:

- Los aviones son más dinámicos y complejos.
- El hardware es mas barato y poderoso
- El software es mas caro y complejo

Estos factores entre otros son los que van dejando en evidencia y plantean la falta de coherencia entre el problema actual y la vieja solución adoptada. Estos factores se ven materializados como problemas evidentes cuando se plantean las siguientes consideraciones:

- Integración desastrosa, ya que el software debía ser re-escrito antes que las pruebas de integración puedan comenzar.
- Una vez integrado no se comportaba como se esperaba debido a que las pruebas no podían comenzar hasta una vez integrado.
- Dificultad de mantener el software, ya que los cálculos y estados estaban dispersos por todos los módulos con los problemas de integridad que esto implica. Además distintos grupos de trabajo obtenían diferentes soluciones para transmisión de estados, etc.
- Interdependencia entre rutinas lo que complica la evolución del simulador.

Las desventajas de este modelo con el pasar del tiempo, motivó la búsqueda de nuevas soluciones, lo cual dio por origen a nuevos modelos estructurales que han sido adoptados en los simuladores contemporáneos. Estos modelos surgidos en la década del 90, serán descriptos a continuación.

## 4.3 - AVSM

### 4.3.1 -Antecedentes y Contexto

La USAF y el SEI han realizado un esfuerzo conjunto cuyo primer objetivo fue la reducción de la complejidad al producir software de simulación de tripulaciones, principalmente respecto a integración. Fue en el año 1986 cuando los ingenieros de la USAF reconocieron que la arquitectura tradicional del software para simuladores de vuelo estaba llegando a sus límites. Anteriormente se utilizaba el diseño por flujo de datos lo que trajo problemas de integración y mantenimiento. Por eso en los últimos 10 años la USAF ha adoptado el uso de Structural Modelling, el cual ha sido aplicado en muchos simuladores con resultados muy exitosos [CHA96][BASS93]. Básicamente ha sido concebido e implementado utilizando como base lenguaje ADA.

El modelo general de simulador de vuelo planteado, trabaja bajo el siguiente esquema [CHA96][BASS93][KAZ01]:

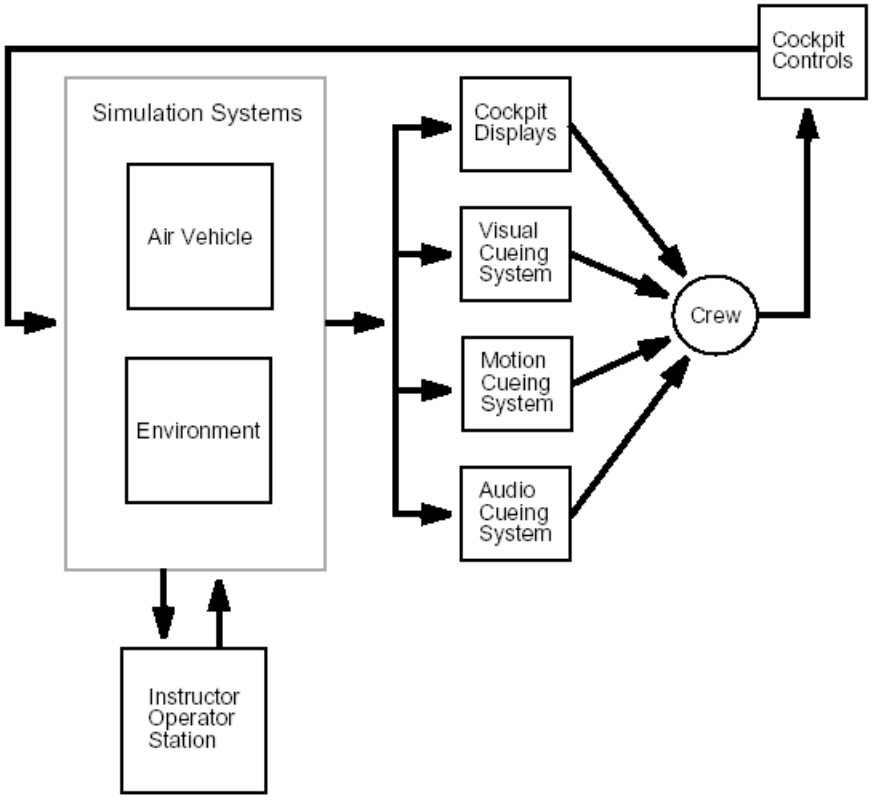


**Figura 3 Modelo de Simulador Genérico**

En este modelo se ve claramente que el corazón de la simulación se representa por el *Simulation Systems* que realiza las funciones propias de la simulación. Estos sistemas que conforman la simulación se hayan divididos en dos partes bien diferenciadas: El *Air Vehicle (AVS)* que se encarga del comportamiento de la aeronave simulada y el *Environment*, encargado del comportamiento del ambiente táctico y natural de la aeronave.

Esta división puede no resultar muy clara ya que por ejemplo si una aeronave arroja una bomba, se presenta la situación que la bomba es parte integral de la aeronave, pero al ser despedida se convierte en parte del ambiente. Entonces, si se modela como una porción de la aeronave que pasa al ambiente, su modelado involucra el paso del control de uno al otro; mientras que si es siempre porción del ambiente, su modelado involucra coordinación entre ambiente y aeronave.

La estación del instructor (*Instructor Operator Station*) típicamente se aloja en un hardware diferente al del modelo de aeronave y alternativamente puede ser alojado junto al modelo ambiental. Esa estación es utilizada para el control y monitoreo de las acciones de la tripulación.

Por último del diagrama, vemos que el Simulador opera como un bucle cerrado en el cual el estado de la simulación se ve alterado por los controles de cabina y/o el instructor, que controla la simulación, mientras que los resultados del nuevo estado de la simulación son presentados en los controles de cabina y/o en la consola del instructor.

### 4.3.2 -Descripción

El AVSM se centró en el problema de modelar el *Air Vehicle System (AVS)*, el cual fue dividido en dos niveles: el nivel superior *Executive*, responsable de las tareas de ejecución en tiempo real y sincronización de la simulación; y la comunicación externa e interna de los estados; y el nivel inferior llamado *Application Level*, que se encarga de calcular el estado de la aeronave simulada. Gráficamente tenemos:

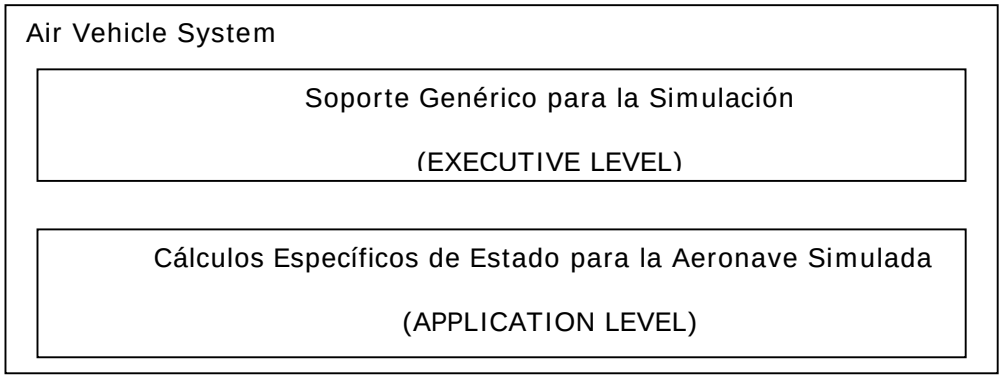


**Figura 4 Air Vehicle System**

Con esto se separa lo que depende de la aeronave simulada de lo común a todas las simulaciones (EXECUTIVE), lo que permite el desarrollo incremental para una aeronave dada y el desarrollo de otros entrenadores reutilizando las partes comunes, lo cual, como se comprenderá es una característica indispensable en los simuladores para diseño de aeronaves. Por otra parte esto permite que los expertos en el dominio se concentren en la parte del software que simule la aeronave y no en lo demás.

Entonces, los cálculos del estado de la aeronave se modelan aplicando una estrategia de dividirla en partes que imitan a las partes reales, teniendo que el comportamiento final resulta de la interacción y relación entre esas partes. Por ejemplo, si tomamos el sistema hidráulico, lo tendremos dividido en Motor de Bomba Hidráulica; Reservorio de Fluido Hidráulico; Válvula de Cortes; etc.; donde cada parte es la menor unidad de cambio, la cual contiene una porción del estado general de la aeronave, limitando los efectos de cambio a calcular e informar su estado.

La mayor diferencia entre el *Executive* y el *Application level* es el nivel de abstracción de sus elementos y el número de instancias. Los elementos del *application level* son totalmente abstractos, o sea existen instancias de ellos y la definición de cada operación se traslada a cada instancia; mientras que los del *Executive* no son tan abstractos.

Presentemos ahora entonces la estructura del AVSM, comenzando por el componente más pequeño (tipo estructural) y ascendiendo dentro de la estructura (comenzamos en el Application Level hasta finalizar en el Executive Level).

### 4.3.3 -Application Layer

Se ha mencionado que el nivel inferior en la estructura del AVSM es el denominado Application Layer. Dicho nivel se encuentra formado por 2 elementos: El Subsystem Controller y el Componente, que se transmiten datos entre instancias del controller y sus componentes.

Cada componente se relaciona con su subsystem Controller(SC) de forma que envía y recibe datos de éste, a la vez que el SC ejerce acciones de control sobre el componente. Esto permite desacoplar al componente de los demás objetos, ya que solo tiene dependencia con su SC. Cualquier modificación o integración es mediada por su controller.

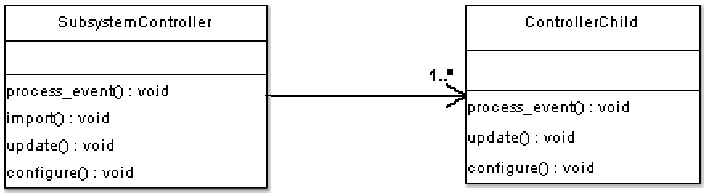


**Figura 5 Application level**

#### *4.3.3.1 Tipo Estructural (Componente)*

El tipo estructural es el elemento en la estructura del simulador de menor nivel que realizan los modelos de simulación de los componentes físicos de bajo nivel. No importa lo que representen, se consideran todos del mismo tipo de modulo. Entonces, representan simulaciones de componentes reales de la aeronave como una bomba hidráulica, un relay, o un tanque de combustible. Pueden soportar modelos específicos de simulador como fuerzas y momentos, pesos, balances, y las ecuaciones de movimiento. Pueden localizar los detalles de equipamiento de cabina como instrumentos, llaves, pantallas. Cada componente provee un algoritmo de simulación que determina su propio estado basado en:

- Estado anterior
- Entradas que representan su conexión con componentes lógicamente adyacentes.
- Un intervalo de tiempo transcurrido

Esta capacidad se llama “updating” y se realiza cada vez que el SC lo solicita. Además del estado normal, un componente debe poder simular mal

funcionamiento a solicitud de su SC, e inclusive poder inicializarse a un valor determinado o una condición conocida.

Todas las interacciones entre componentes son mediadas por el SC, que posee el conocimiento para usar las operaciones de los componentes para lograr los requerimientos del subsistema con un todo. Esto incluye:

- Propagar periódicamente los cambios de estado a los componentes (update)
- Realizar conexiones lógicas entre los componentes usando parámetros de entrada y salida en esas operaciones.
- Realizar conexiones lógicas entre componentes y el resto de la simulación usando las conexiones internas y externas.

Los malos funcionamientos de los componentes se asumen asociados a una operación anormal del componente del mundo real que se esta modelando. Por lo tanto la presencia de tales son decididas por el diseñador del componente que las expone al diseño del SC para que pueda realizar los pedidos de mal funcionamiento. Es responsabilidad del SC mapear entre fallas del subsistema y la de los componentes.

El tiempo en que debe calcular su estado y la comunicación del mismo es controlado externamente, por lo cual se definen las siguientes interfaces que serán compartidas por todos los componentes:

- *Update (periódica)*: basada en el estado actual y la entrada calcula el próximo estado. Los parámetros de entrada contienen toda la información necesaria para este cálculo.
- *Configure (aperiódica):* Fija el estado interno del componente basado en los parámetros de entrada.
- *Process_event (aperiódica):* altera el estado interno basado en los parámetros de entrada (por ejemplo fallas).

Basado en esto, cada elemento estructural expone la misma interface hacia su Subsystem Controller, entonces puede ser representado en forma general como:

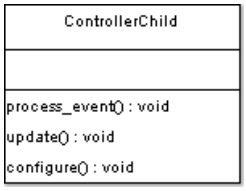

**Figura 6 Tipo Estructural**

#### *4.3.3.2 Subsistemas*

Cada susbsistema, por fuera, es el agrupamiento de las partes según el subsistema de la aeronave que representan y actúan en conjunto para afectar el comportamiento de la misma (Por ejemplo Sistema Hidráulico). Quién realiza este agrupamiento es el denominado *Subsystem Controller (SC)*, el cual es un componente que agrupa y controla los componentes relacionados dentro del subsistema. El SC será el único componente que podrá interactuar con el interior y el exterior de un subsistema.

Por lo tanto los SC se utilizan para servir como interfaz entre los componentes que encapsulan y el sistema, representando la simulación de un subsistema como un todo y realizan el control y comunicación no periódico entre el sistema y los subsistemas.

Debido a que el Modelo restringe la comunicación entre componentes, un SC debe proveer la capacidad de hacer conexiones lógicas entre sus componentes y los de otros subsistemas. Son responsables de determinar como usar las capacidades de sus componentes para satisfacer las funcionalidades del entrenador como son malfuncionamiento o establecer parámetros.

En este nivel, entonces vemos que el AVS está formado por un Executive y varios subsistemas.

Toda la comunicación del estado de un subsistema y la ejecución del cálculo del mismo están dirigidos externamente al subsistema. Por lo tanto cada SC expone una serie de métodos necesarios para posibilitar estas operaciones:

- *Update (periódica)* basada en el estado actual calcula el próximo estado. Se diferencia del componente ya que no se le pasan parámetros de entrada, sino que los datos son comunicados utilizando un single write – multiple readers de las areas de datos exportados implementado por memoria compartida. Los SC reciben el estado de la simulación desde el exterior. El estado *operate* lo instruye a ejecutar las operaciones de sus componentes en algún orden lógico, de forma que los cambios se propaguen entre ellos, avanzando el

estado de la simulación. Es más que un secuenciador, sino que provee la unión lógica para el paso de datos, transformaciones y conversiones. El estado stabilize indica al SC que finalice su computación de una manera controlada (por ejemplo para prevenir el movimiento de la plataforma de forma que produzca daños a la tripulación), de la siguiente forma: (1) Obtener y almacenar localmente los valores de las conexiones internas bajo el control directo del executive. Esto es para lograr temas de consistencia de datos y coherencia de tiempo. (2) Estabilizar los algoritmos de simulación de sus componentes y reportar cuando considera el subsistema como un todo actualmente estable. Update tiene un parámetro de salida para indicar cuando se considera que el subsistema se encuentra estable.

- *Configure (aperiódica):* Se utiliza en estados como initialize los cuales son predominantemente aperiódicos. Se utiliza para establecer un conjunto de condiciones como es la configuración de un dispositivo o misión. Para completar la operación el SC invoca operaciones de sus componentes que causan que el componente establezca estas condiciones.
- *Process_event (aperiódica):* Es usado en estados operativos que son predominantemente periódicos para pedirle al SC que responda a un evento. El evento es pasado como parámetro a la operación. Hay muchos eventos como los de la estación de instructor (por ejemplo introducir fallas).
- *Import (periódica):* hace que el SC acceda a las áreas de datos de otros subsistemas para obtener los datos necesarios para sus cálculos. Este método se utiliza para evitar la colisión de datos al acceder a áreas de memoria compartida y es ejecutada externamente por el Executive.

La representación es:

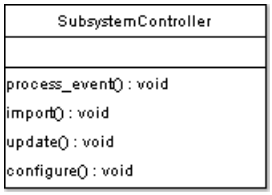


**Figura 7 Subsistema**

Este modelo provee una encapsulación común para la vista del Executive, permitiendo simplificar las tablas de scheduling.

Por dentro del subsistema, las tareas del SC son:

- Invoca sus componentes
- Brinda información de estado a sus componentes
- Recibe el estado desde sus componentes.
- Restablece el estado de sus componentes (Por ejemplo cuando. el instructor. reinicia el ejercicio).
- Altera el estado de sus componentes en forma no periódica (Por ejemplo cuando se introduce una nueva mal función para el ejercicio).

La información de estado que ingresa/sale se mantiene en el SC como variables locales.

### 4.3.4 -Executive Layer

Hasta el momento hemos descrito el nivel de aplicación que está formado por los subsistemas. Continuando el ascenso dentro de la jerarquía estructura, ingresamos al Executive Layer el cual es una capa genérica a todas las simulaciones, ya que se encarga de la ejecución en tiempo real, comunicaciones y la sincronización de la simulación.

Los elementos constituyentes del nivel ejecutivo se muestran en la siguiente figura y a continuación procederemos a describirlos.

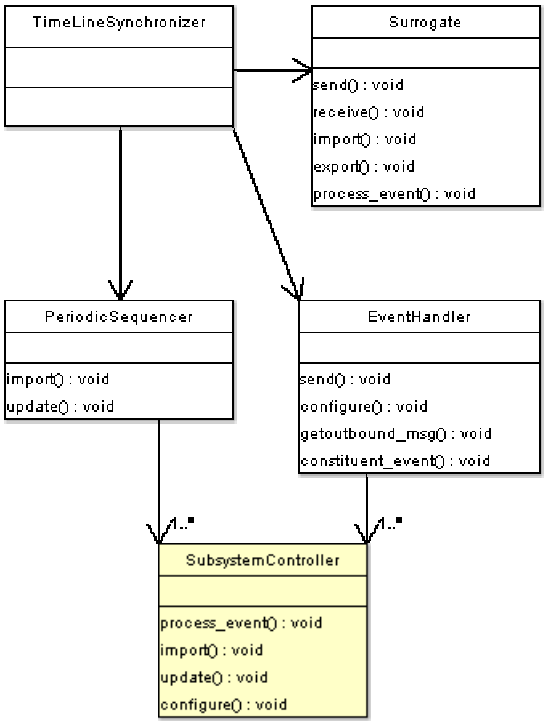


**Figura 8 Executive Level**

#### *4.3.4.1 Timeline Synchronizer (TLS)*

El TLS es el mecanismo base de scheduling del modelo de aeronave que se encarga de mantener la noción de tiempo interno de la simulación y su estado. Pasa y recibe, datos y control con los otros 3 elementos del Executive y además coordina el procesamiento periódico y aperiódico con otros procesadores que pueden poseer su propio TLS. En definitiva es un Executive de simulación cíclica.

En un intervalo de tiempo (ciclo) la ejecución del TLS implica:

- Acceso a datos
- Procesamiento Periódico
- Procesamiento de Eventos
- Sincronización interproceso

Comienza el intercambio sincronizado entre las áreas de exportación de datos de los SC (utilizando el método data_moves) del PS, e invoca al PS y al EH en tiempos dados durante un frame. En una configuración multiproceso, un proceso asume la función de master TLS y corre en la mayor

frecuencia que los otros procesos, utilizando comunicación interproceso (por ejemplo con semáforos) para coordinar el inicio de los frames, y períodos dentro del frame en todos los procesos.

#### *4.3.4.2 Periodic Sequencer (PS)*

El PS conduce todo el procesamiento periódico realizado por los subsistemas. Involucra invocar los subsistemas para realizar operaciones periódicas acorde a schedules fijos. Es el encargado de llamar a los métodos periódicos de los SC.

Provee 2 operaciones al TLS:

- *Import* solicita que el PS invoque a los subsistemas para realizar su operación de import.
- *Update* que llama las correspondientes update de los subsistemas.

#### *4.3.4.3 Event Handler (EH)*

El EH se encuentra en el mismo nivel que el PS y se utiliza para ejecutarr el procesamiento aperiódico de los subsistemas.

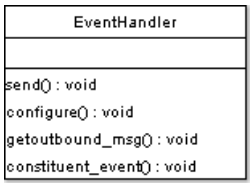


**Figura 9 Event Handler**

Su propósito es servir de interfaz con la consola de instructor. Esta consola envía eventos a una cola indicando el comienzo o final de mal funcionamientos. El EH toma estos eventos como consecuencia a un llamado a *event_processing* y despacha el control al SC apropiado según una tabla de despacho acorde al evento procesado. La información de a cual SC llamar y qué operación debe solicitar se pasa como parte de los parámetros de invocación del método. Por ejemplo al recibir un evento de detener la simulación, en la tabla de despacho figurará comunicar al PS el estado general de la simulación.

Provee 4 operaciones al TLS:

- *configure*: usado para comenzar una nueva misión.
- *constituent_ event:* usado cuando un evento esta destinado para una instancia de un modulo
- *get_outbound_msg:* usado por el TLS para conducir procesamiento aperiodico en estados que son periodicos como operate.
- *send:* usado por SC para enviar eventos a otros SC y mensajes a otros sistemas.

Para su procesamiento el EH requiere: determinar que SC recibe un evento, usando conocimiento del mapeo entre identificadores de eventos e instancias de subsistemas. Luego se debe invocar los subsistemas y extraer los datos requeridos de los eventos antes de la invocación.

#### *4.3.4.4 Surrogate*

El Surrogate, es quién provee la comunicación periódica externa. Intermediario responsable de la comunicación system-to-system entre modelo de aeronave y ambiente o instructor. Se encargan de los detalles físicos del sistema con el que se comunican, y son responsables por representación, protocolo de comunicaciones y demás. Trabaja por pares: por cada sistema externo que este sistema se comunica, hay un surrogate en el sistema local y uno en el externo, ocultando los detalles de conexión como son diferentes representación de datos entre procesadores.

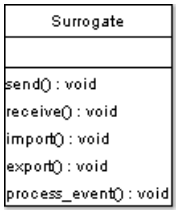


**Figura 10 Surrogate**

Por ejemplo: el surrogate toma los datos y los envía a la estación del instructor. Si el instructor desea fijar un estado para la tripulación, el evento es recibido por el surrogate y pasado al EH para despacho al subsistema apropiado.

Esto permite que el PS y el EH se mantengan ignorantes de los detalles de la estación de instructor y el modelo de ambiente. Todo se encapsula en el Surrogate y cualquier cambio no se propaga más allá de este.

Las operaciones que expone son:

- *Send*: envía el estado a su par en otro sistema
- *receive:* acepta el estado proveniente de otro sistema
- *export:* hace disponible el estado recibido al sistema
- *import:* instruye al surrogate para preparar la información deestado para ser enviada.

#### *4.3.4.5 Flujo de Control*

Entonces tenemos que la coordinación y comunicación entre elementos se hace de la siguiente forma:

1. Entre SC: utilizando export areas
2. Entre instructor y el EH: utilizando la cola de eventos.
3. EH y PS: utilizando una variable que indica el estado general de la simulación
4. master TLS - TLS: mecanismo de comunicación interproceso (por ejemplo Semáforos).

Analizando el flujo de control, tenemos un modelo Top-Down. Por ejemplo el TLS llama al PS pero nunca al revés. La figura siguiente muestra el flujo de control durante un update periódico (cada Subsystem controller es llamado en paralelo, o sea representan varios procesos simultáneos):

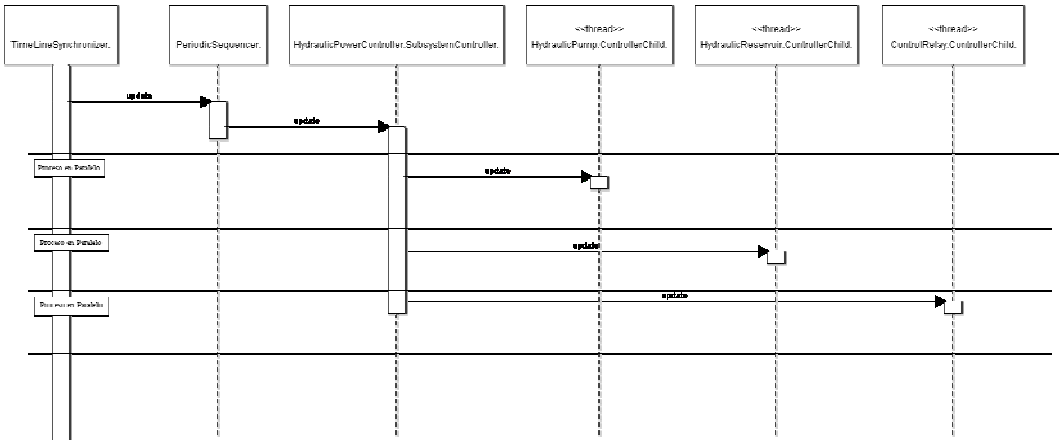


**Figura 11 Flujo de Control Periódico**

En la figura siguiente se muestra el flojo de control en respuesta a un evento:

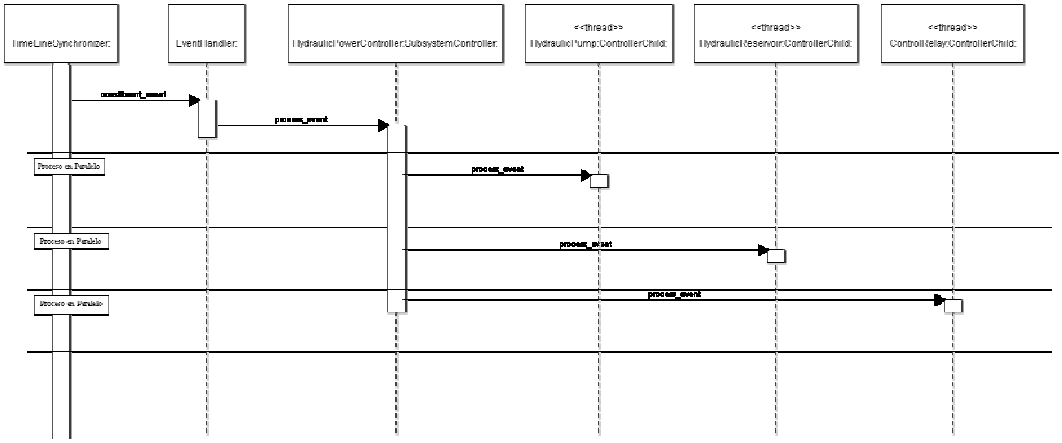


**Figura 12 Flujo de Control por evento**

En ambas figuras los TLS, PS y EH son instancias que corren en procesadores específicos.

### 4.3.5 -Uso del Modelo

Hasta aquí hemos descrito la arquitectura del sistema, o sea un framework estructural para el simulador, pero sin detalles funcionales. Es decir que la descripción del modelo estructural puede utilizarse para un helicóptero como para un reactor nuclear.

El próximo paso es darle funcionalidad a la arquitectura con subsistemas apropiados para cada tarea. Tenemos la arquitectura definida por 6 módulos: Componente, SC, TLS, PS, EH y Surrogate. Esto hace una estructura comparativamente simple de construir, comprender, integrar, ampliar y modificar.

Para definir la funcionalidad y los elementos intervinientes, se comienza definiendo instancias de los SC de forma de detallar las particularidades de la aeronave a simular. La partición se hace acorde a los sistemas y complejidad de la aeronave como también de los tipos de entrenamiento para los cuales se diseña el simulador. Por ejemplo típicamente se comienza formando grupos, que típicamente serán:

- *Cinemática:* elementos que tratan con fuerzas ejercidas sobre la aeronave.
- *Sistemas del Avión:* partes realacionadas con sistema comunes que proveen a la aeronave de potencia o que distribuyen la energia dentro de la estructura.

- *Aviónica:* Cosas que proveen algún tipo de soporte a la aeronave pero no tienen que ver con la cinemática, el control o la operación de un sistema básico de vuelo (ejemplo: radios).
- *Ambiente:* cosas asociadas con el ambiente en el cual el vehículo opera.

Luego esos grupos se dividen en sistemas y estos en Subsistemas. Como se puede deducir, los grupos y sistemas no se refeljan directamente en la arquitectura – no hay un controlador de grupo – y existen para organizar la funcionalidad asignada a las varias instancias de los SC.

Luego particionar los modelos matemáticos y físicos (aerodinámica) es mucho más dificultoso que particionar la estructura física de la aeronave.

Entonces se dividen los grupos en sistemas, que es una solución autocontenida a problemas de simulación. Estos generalmente se reflejan como módulos de código implementados por un grupo de ingenieros. Entonces tomando el grupo cinemática, tendremos los elementos relacionados al control del movimiento del vehículo y la interacción del vehículo con sus superficies de control con el ambiente. Los sistemas serían:

- Estructura
- Propulsión
- Tren de aterrizaje
- Controles de Vuelo

El paso siguiente sería tomar cada sistema y obtener los subsistemas correspondientes. Por ejemplo el sistema de propulsión tratará con los motores de la aeronave. Estos motores serán manejados creando múltiples conjuntos de variables de estado e instancias a objetos que permitan calcular los momentos de rotación del motor; fuerzas y momentos causados por la distribución de la masa de combustible, etc. El subsistema de combustible se ubica aquí porque su interfaz primaria es con los motores y se encarga de calcular las fuerzas actuantes en la estructura proveniente del movimiento de combustible dentro de los tanques, como también del efecto gravitacional de la masa del combustible.

Hasta aquí hemos identificado la división de funcionalidad, la ubicación de los subsistemas y SC, y las conexiones entre subsistemas. Para completar la arquitectura necesitamos lo siguiente:

- Identificar los componentes para el subsistema de propulsión

- De la misma forma descomponer en otros grupos, sus sistemas y subsistemas.

## 4.4 - DARTS

### 4.4.1 -Antecedentes y Contexto

Independiente al desarrollo del AVSM, Boeing desarrollaba el proyecto de simulador modular denominado Mod Sim. Con este antecedente y la influencia del AVSM, desarrollaron la denominada Domain Architecture for Reuse in Training Systems (DARTS) como un esfuerzo de investigación y desarrollo interno, la cual fue utilizada en el programa Software Technology for Adaptable, Reliable Systems (STARS) del Department of Defense Advance Research Projects Agency (ARPA).

En resumen, Boeing intentó unir el reuso de Structural Modelling (en particular del AVSM) y la experiencia obtenida en Mod Sim.

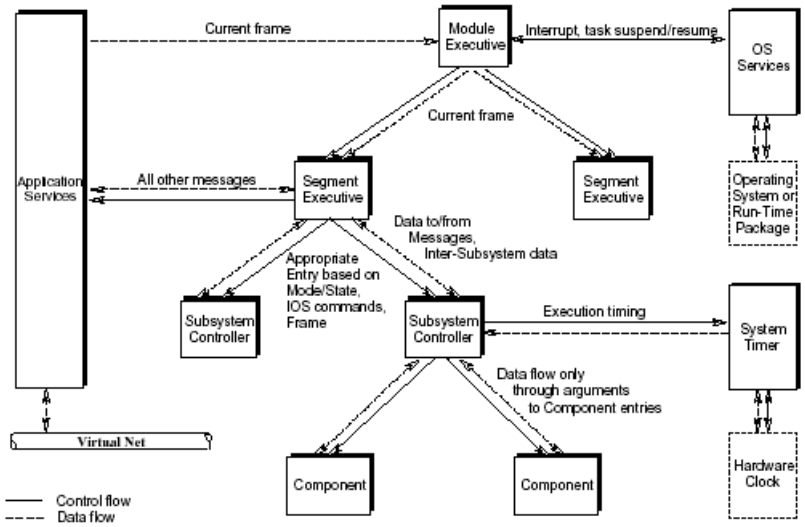


**Figura 13 DARTS**

La característica de DARTS es que representa un simulador genérico que puede ser adaptado a cualquier simulador presente o futuro, tal cual lo ha demostrado en la práctica [CRI93].

DARTS ha sido aplicado en el T-34C Flight Instrument Trainer y se encuentra relacionado con la teoría denominada Megaprogramming que se

define como “la práctica de construir y evolucionar software de computadora componente por componente”.

### 4.4.2 -Descripción

DARTS aplica el AVSM dentro de cada módulo y segmento. Aquí, un módulo es un sistema computacional que tiene un Executive, el cual contiene todas las funciones de interrupciones, suspensión de tareas, etc. Estos se encargan de ejecutar a los segment Executives.

En DARTS el Air Vehicle System, se encuentra dividido en 125 funciones, o áreas de capacidad. A su vez se encuentra dividido en doce segmentos y cada función de las anteriores es asignada a un segmento. Por lo tanto gráficamente se tiene:

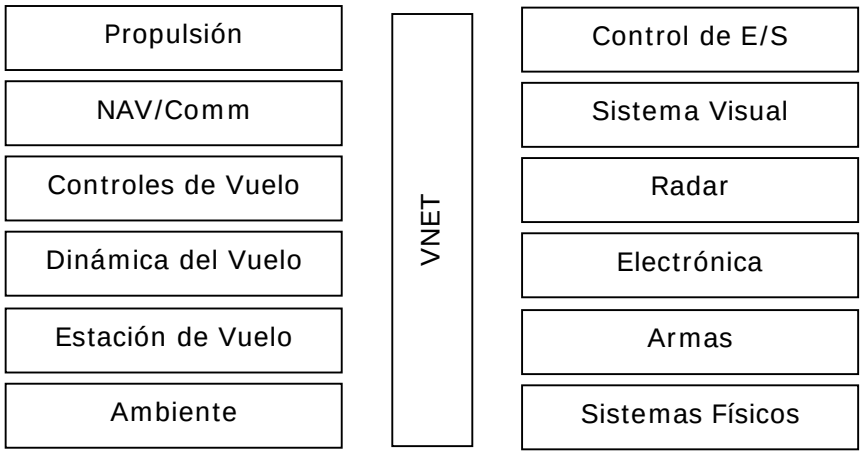


**Figura 14 Segmentos en DARTS**

Cada segmento se caracteriza por coherencia interna y poco acoplamiento externo, o sea las funciones en un segmento se relacionan por flujo de datos, orden de ejecución o dependencia. Los segmentos tienen interfaces definidas y son las únicas que permiten comunicación entre segmentos.

Entonces tenemos que el modelo en DART está formado por 5 componentes: Virtual Network (VNET), Module Executive(s), Segment Executive, Subsystem Controllers y Componentes. En general entonces tenemos:

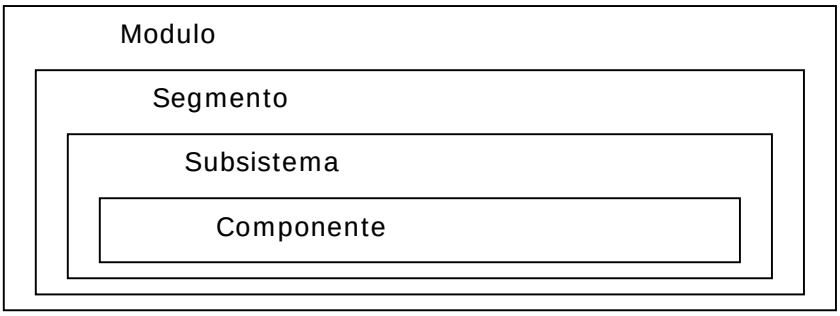


**Figura 15 Estructura de DARTS**

Al igual que AVSM los componentes obtienen sus datos a través de llamadas a funciones en vez de leerlos desde memoria compartida. Esto disminuye la dependencia entre objetos y hace más fácil el reuso. Cada mensaje se define como un tipo de Ada y se agrupan en paquetes en concordancia al segmento que puede emitir ese mensaje. Además, se definen los receptores y emisores legales para un mensaje, de forma que ningún otro segmento pueda emitir o recibir ese tipo de mensajes. Todos los segment executive se comunican utilizando la VNET, lo que hace que los elementos inferiores sean más reutilizables.

La diferencia en el SC en DART es que los datos entre Subsistemas son transmitidos a través de una VNET que posee la inteligencia suficiente para proveer servicios de forma tal que el componente que se trate no necesita conocer sobre le hardware en que corre.

Otra diferencia es el segment executive que agrupa componentes y subsistemas en común, generalmente referidos a componentes que poseen mucho intercambio de datos entre ellos. Los segment executives se comunican entre ellos utilizando la VNET, ya que es el medio para comunicarse con otros componentes y subsistemas localizados en otros segmentos.

El Module executive es responsable de asignar recursos y comunicación entre procesadores. Su ejecución puede ser por llamada a funciones o activando procesos en pausa, dependiendo de la implementación.

Los application services se ocupan de los detalles de implementación ya que por ejemplo oculta detalles tales como si la comunicación se realiza utilizando memoria compartida, una red Ethernet, etc.

### 4.4.3 -VNET (Virtual Network)

Este componente ha sido incorporado por DARTS y es la mayor diferencia con AVSM. Representa la estructura de comunicaciones entre segmentos. Se definen mensajes (estructuras ADA), y una tabla de receptores y emisores para ese mensaje.

VNET provee la interfaz de comunicación, ocultando la implementación para la transmisión de mensajes, haciendo este mecanismo independiente de la plataforma.

Como se menciona en [CRISPEN01] esta aproximación posee muchas ventajas comprobadas, mientras que la única objeción es la performance frente a memoria compartida, situación que, a medida que crece el poder de procesamiento se hace menos evidente.

### 4.4.4 -Module Executive

Un módulo es un sistema computacional con sus dispositivos asociados o bien un procesador en un sistema multiprocesador. Su propósito es ejecutar los elementos de menor nivel.

Los module executives causan que se ejecuten los segment executives, lo cual puede realizarse como llamadas a subprogramas o bien tareas independientes, y la comunicación consiste simplemente en pasar los pulsos de reloj.

### 4.4.5 -Segment Executive

Implican un grupo funcional, ya que entre los componentes hay mucho intercambio de datos u orden de dependencias. Son responsables de la comunicación sobre VNET y se encargan de llamar las entradas de los SC ante eventos aperiódicos. La diferencia entre DARTS y AVSM es que las funciones como cambio de modo o bien fallas, se manejan como mensajes comunes procesado por el segment executive (no hay event handler como en AVSM).

### 4.4.6 -Subsystem Controller

Están implementados como en el AVSM. La diferencia radia en que en AVSM los datos salen del subsistema utilizando áreas de memoria compartida, mientas que en DARTS, el segment executive provee al SC con datos y construye los mensajes para enviarlos a la VNET. Toda la comunicación entre

subsistemas toma lugar utilizando buffers que mantiene el segment executive.

El método initalize de cada componente provee los datos de entrada y salida, por lo cual, una vez ejecutada puede ejecutarse sin error aunque no haya datos provenientes de la VNET. Esto evita el problema sobre los componentes o subsistemas que necesitan ejecutarse antes que otros para evitar que datos erróneos se procesen. VNET provee información en la función import para determinar si se han recibido datos desde la última iteración. DARTS copia los datos al buffer sólo cuando datos nuevos han sido recibidos.

### 4.4.7 - Component

Al igual que en AVSM, es el menor elemento estructural y se encuentra implementado de la misma forma, representando un elemento físico de la aeronave o bien presiones, flujos, etc. Por lo tanto cada componente puede computar su estado de manera abstracta y por lo tanto reutilizable.

## 4.5 - Otras Implementaciones

SAAB posee varios simuladores desarrollados implementando AVSM los cuales aplican variaciones al modelo acorde a las necesidades particulares del caso, los cuales se describirán brevemente a continuación.

### 4.5.1 -PMSIM

Se utiliza para el desarrollo de sistemas de control de vuelo; estudio de interface Hombre-Maquina; sensores de vuelo y sistemas de armas. Todos los sistemas son simulados por software.

Aquí la frecuencia que utiliza el Executive se obtiene de un circuito externo de reloj.

El Executive corre a una frecuencia de 60 HZ provenientes de hardware real de la aeronave, que se usa para sincronizar el modelo de software con el de hardware. En realidad posee varios schedulers a 1, 7.5, 5, 15, 30 y 60 HZ que conocen el orden de ejecución de sus modelos, lo que asegura que cada modelo se ejecuta con el input correcto. Aquí se encuentra uno de los problemas de estas implementaciones: utilizan memoria

compartida para transmitir datos entre los modelos, lo cual no garantiza que cada modelo este utilizando los datos correctos.

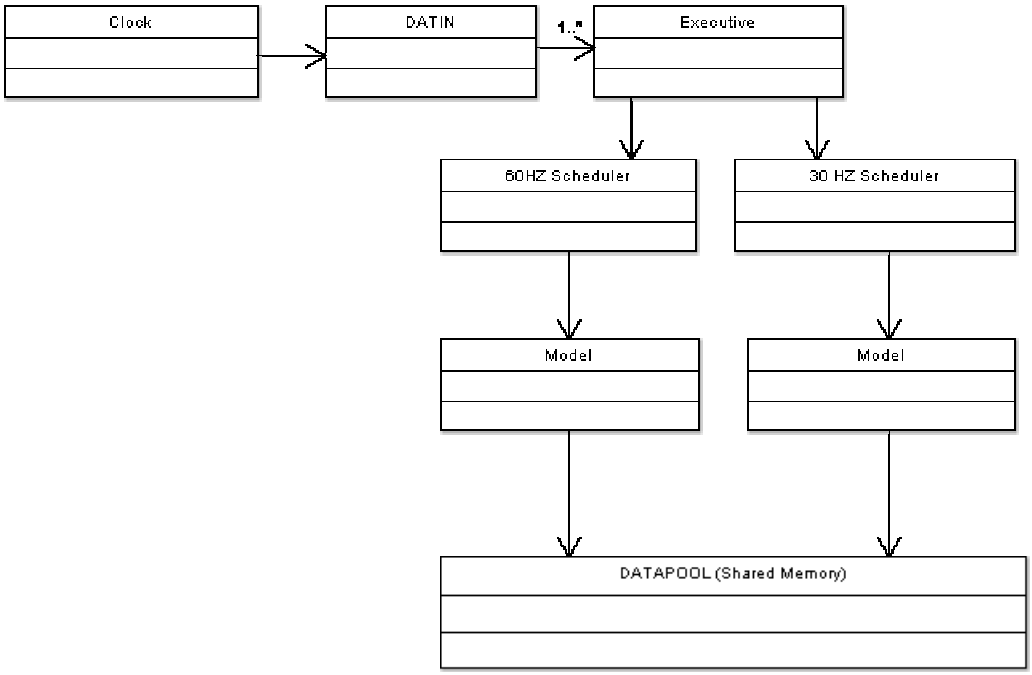


**Figura 16 Estructura de PMSIM**

### 4.5.2 -SYSIM

Se utiliza para el desarrollo de incluyendo el hardware y software que irá en la aeronave. En esta implementación tenemos una mezcla de sistemas reales (hardware real) y sistemas simulados por software. El uso de sistemas reales se vuelve mandataria cuando los modelos de software no pueden reproducir las condiciones de tiempo necesarias.

### 4.5.3 -FMS (Full Mission Simulator) y MMT (Multi Mission Trainer)

Se utilizan para el entrenamiento de pilotos. En este caso constan de varios computadoras/procesadores comunicados por memoria compartida y red ethernet para la consola del instructor.

Básicamente la sincronización de datos se realiza por las etapas que contiene un ciclo de ejecución. En la etapa export se escriben los datos, que interesan a los otros procesadores, en la memoria global. Update procesa y escribe los datos en la memoria local de cada procesador y Event es donde se verifica si existen eventos pendientes de ser procesados.

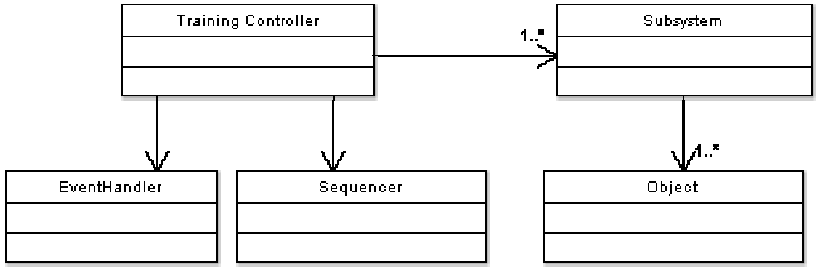


**Figura 17 Estructura de FMS/MMT**

## 4.6 - Enfoque Orientado a Objetos

Como se referencia en [CRI93] y como se ha graficado para algunos casos en la presentación de AVSM, los componentes de DARTS pueden asimilarse como objetos, pero como se basa en una descomposición funcional, ocasionalmente un componente es dividido en varios segmentos. Por ejemplo tenemos la bomba de combustible en el subsistema Propulsión, pero para producir el sonido de la bomba de combustible se tiene un componente en el subsistema de Audio. Por ello los creadores de DARTS han preferido utilizar el termino Object Abstracted Design.

Generalizando, el modelo estructural reconoce la existencia de objetos, evidenciado principalmente en los componentes, pero no ha sido desarrollado utilizando Análisis Orientado a Objetos.

Para recalcar las similitudes del modelo estructural con el Análisis Orientado a Objetos, podemos mencionar que es un proceso de desarrollo iterativo e incremental, en el cual se hace uso de la abstracción, encapsulación, modularidad y tipos organizados en jerarquías, existiendo una correspondencia entre clases y tipos estructurales.

Como mencionamos en su momento, el modelo estructural ha sido desarrollado en el contexto de Ada como lenguaje de implementación, por lo cual tiene la característica de Ada que, por lo menos hasta ese momento, está basado en objetos pero no orientado a objetos (la diferencia radica en el soporte de herencia). Por otra parte en vez de tener clases abstractas, posee tipos estructurales y las no existen como tales clases concretas que podrían capturar las propiedades comunes a los grupos de componentes similares.

El modelo estructural ha demostrado sus ventajas y es por eso que el objetivo de esta tesis es mantener estos beneficios pero agregando las cualidades del Análisis Orientado a Objetos y poder extraer Patrones de Diseño para documentar el conocimiento contenido en estas soluciones de diseño. Además la aplicación de Patrones de Diseño existentes para mejorar la estructura y la reutilización basada en soluciones probadas.

A primera vista los beneficios inmediatos de utilizar Análisis Orientado a Objetos es que el polimorfismo permite refinar los métodos heredados y el código cliente actúa sobre una colección de objetos heterogéneos como una colección de la clase base. Esto se evidencia sobre todo en la diferencia que los lenguajes procedurales tratan los datos y funciones por separado, formando un programa que debe tener estructurado de antemano, cuales funciones actúan sobre que dato y cuándo. Por lo tanto tienen un tipo fijo de vehículo, en el cual se cambiarán las tablas u otros componentes cada vez que se quiera simular otro modelo de aeronave. Con el modelo Orientado a Objetos se podría explotar las funcionalidades comunes permitiendo un mayor reuso.

## 4.7 - LaSRS++

Antes de continuar, haremos una breve descripción sobre Langley Standard Real-Time Simulation in C++ (LaSRS++) framework, que es un antecedente que debe ser mencionado al trabajar sobre este enfoque orientado a objetos.

LaSRS++, no se encuentra basado en el modelo estructural, sino que surge como un esfuerzo realizado en 1995 para migrar el LaRC (real-time simulation environment) realizado en lenguaje FORTRAN procedural estructurado en más de seis repositorios para realizar tipos específicos de investigación utilizando diferentes cabinas de simulación.

Algunas de las motivaciones para el cambio a C++ fueron, la complejidad de la curva de aprendizaje entre proyectos; dificultad de balancear la carga de trabajo entre equipos de desarrollo; problemas de modificación y pruebas para pasar una aeronave de un repositorio a otro y sólo se podía simular un vehículo por vez (no era multi-vehículo).

Utilizando los conceptos de abstracción, Encapsulación, herencia y polimorfismo provistos por la tecnología Orientada a Objetos se obtuvo el siguiente framework:

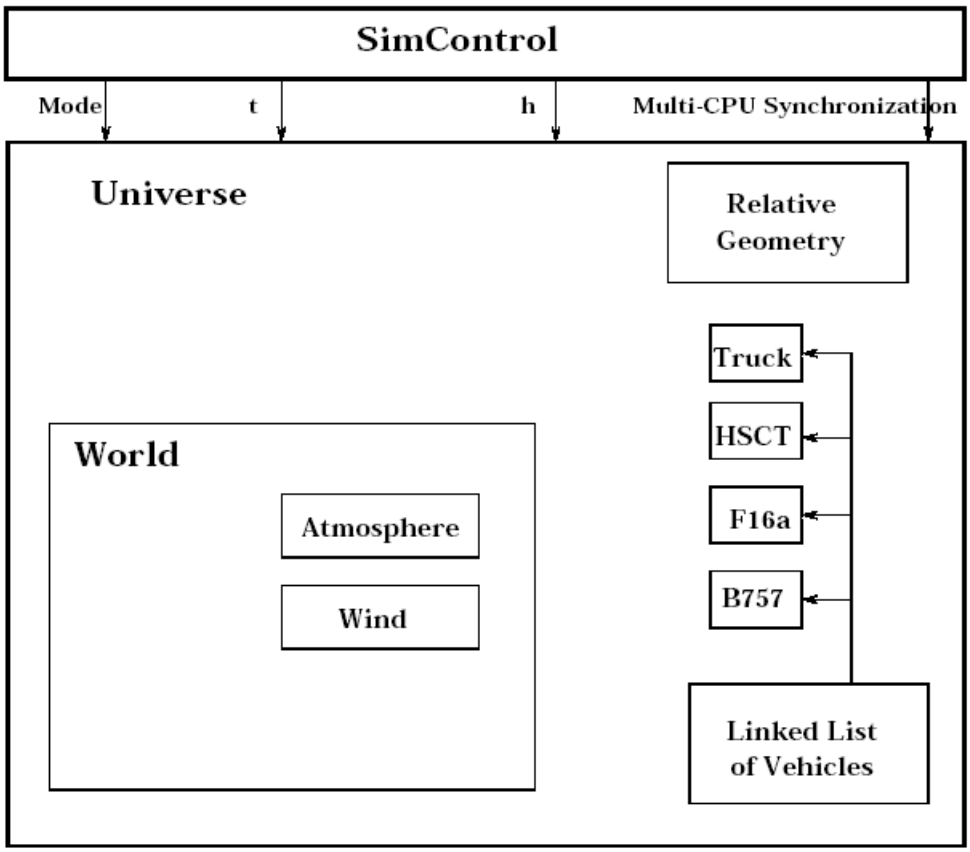


**Figura 18 Modelo Conceptual de LaSRS++**

Todas las aeronaves derivan de un conjunto de clases abstractas que proveen la interface del framework:

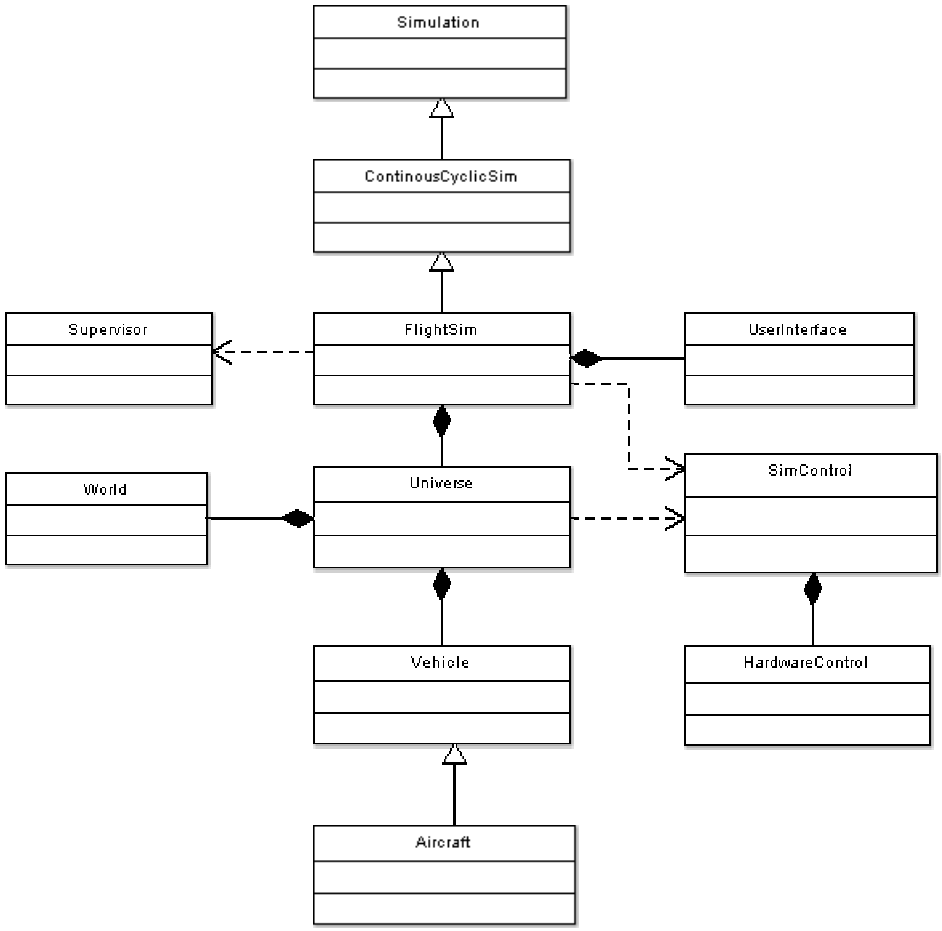


**Figura 19 Vista de Nivel Superior del Framework**

Las funciones de cada una de estas clases son:

- *Simulation:* Provee una interface abstracta para permitir a la simulation concreta poseer el método execute().
- *ContinuousCyclicSim:* Agrega una interface abstracta que soporta los conceptos de comportamiento dependiente del modo y tiempo que corre en intervalos discretos.
- *FlightSim:* Define la inicialización y ejecución de los vehículos dinámicos.
- *Universe:* Provee el ambiente en el cual los vehículos se ejecutan.
- *World:* Provee un mundo para que los vehículos recorran. Encapsula los vientos y la atmósfera.
- *Vehicle:* Es la abstracción del vehículo dinámico.
- *Aircraft:* Agrega el comportamiento específico a un aeroplano.

- *SimControl:* Provee la información sobre tiempo y sincronización.
- *HardwareControl:* Interface abstracta al hardware usado en la simulación.
- *Supervisor:* Provee sincronización con el reloj de tiempo real.

Al igual que en el modelo estructural se mantienen mínimas las interfaces, tratando de generalizar métodos como doOperate(), situación que se grafica a continuación:

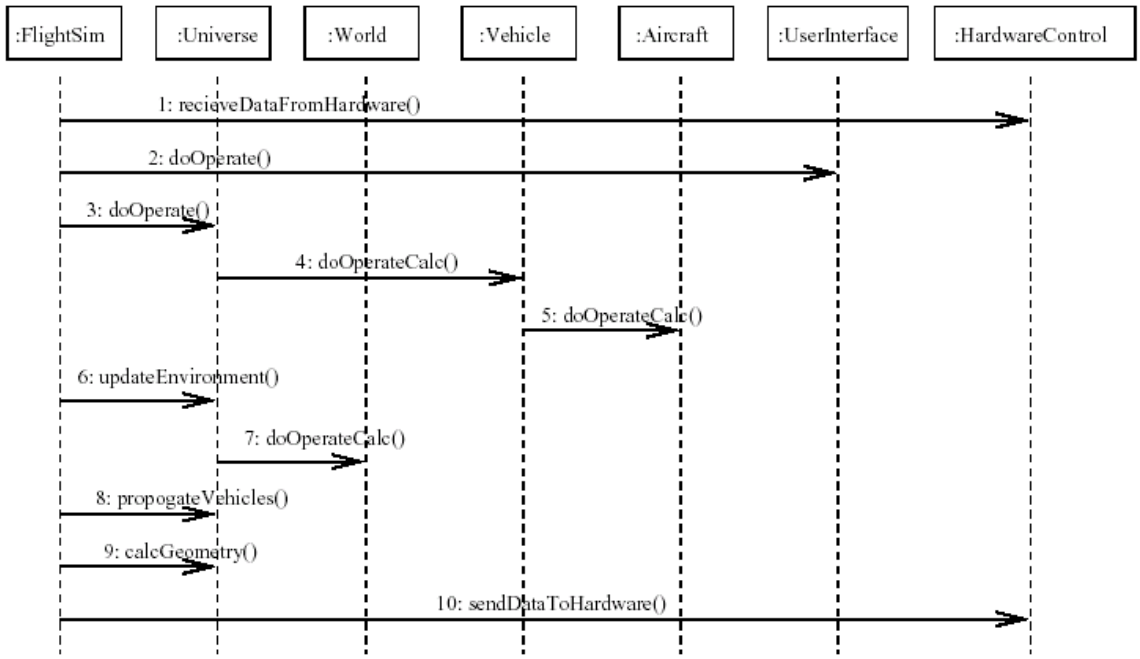


**Figura 20 Interacción entre objetos**

Soporta N vehículos corriendo en M procesadores, pero la única limitación es que los componentes del vehículo deben residir en la misma CPU.

Una particularidad digna de mencionar es que los sistemas del Boeing 757 han sido desarrollados utilizando este framework y fueron movidos a la aeronave sin realizar mayores modificaciones. Este concepto denominado sim-to-flight ha sido facilitado por el uso de la tecnología orientada a objetos para desarrollar el código.

Para resumir, se ha desarrollado un framework utilizando análisis Orientado a Objetos implementado en C++, lo que resulta en un sistema

flexible capaz de simular múltiples instancias de objetos dinámicos distribuidos en varias CPU's.

## 4.8 - Conclusiones

Se observa que AVSM separa lo que depende de la aeronave simulada de lo común a todas las simulaciones (EXECUTIVE), lo que permite el desarrollo incremental para una aeronave dada y el desarrollo de otros entrenadores. Esto es una ventaja para nuestro tipo de simulación ya que el modelo de aeronave debe cambiar diseño a diseño.

Ambos modelos proveen una descomposición jerárquica en bloques que imitan la división existente en la aeronave (subsistemas y componentes). Para lograr el reuso, limitan las interfaces. La comunicación en AVSM se lleva a cabo de forma muy rápida por llamadas a funciones y memoria compartida, pero a la vez esta forma se torna muy compleja cuando hay muchos datos en juego y la comunicación entre subsistemas, cuando, por ejemplo es necesario agregar uno nuevo.

DARTS posee una comunicación más lenta pero tiene la ventaja de centralizar la información, la cual puede ser perfectamente controlada a través de la VNET.

Todos estos modelos buscan obtener un diseño con la mejor relación entre performance, integrabilidad y modificabilidad, mediante la restricción de los tipo de configuración; comunicación entre modulos y la descomposición de funcionalidad para anticipar los cambios de la aeronave simulada.

Las mejoras en estos simuladores se deben a una mejor comprensión e incorporación de una arquitectura de software bien documentada y analizada.

Chastek y Borwnsword describen algunos de los resultados obtenidos a través del uso de este modelo. En un data-driven simulador previo de tamaño comparable (B-52), se identificaron 2000-3000 problemas de testeo se identificaron durante los test de aceptación. Con su proyecto de modelado estructural, se reportaron 600-700. Encontraron los problemas mas fáciles de corregir; muchos surgieron de malos entendidos con la documentación. El personal puede separar un problema offline en vez de tener que hacerlo en el lugar. Desde el uso del modelado estructural, la tasa de defectos para un proyecto es la mitad que en los utilizados en los simuladores data-driven.

La performance de tiempo real se obtiene mediante la operación del executive y el uso del PS. Obviamente involucra varios procesadores y el tiempo real proviene de que todos los ciclos de control se encuentren dentro de un período del simulador.

La integración se logra gracias a que las interfaces se encuentran minimizadas. Para un subsistema a través de su SC, lo que hace más fácil agregar nuevos componentes. El hecho de que el SC se convierta en el conducto de datos, agrega complejidad y hace caer la performance, pero en la práctica es mayor el beneficio que se vincula a esto como el poder construir un sistema "esqueleto" para realizar el desarrollo incremental y fácil integración.

La modificabilidad se logra ya que existen pocas configuraciones por lo cual son fáciles de entender, y al tener la funcionalidad localizada se puede trabajar sobre los subsistemas involucrados ante una modificación en particular. Recordemos que los sistemas siguen el modelo real por lo cual es más fácil realizar estos cambios ante un cambio en la aeronave real.

## 4.9 - Temas a Tratar

Hasta el momento hemos realizado una revisión de la bibliografía para brindar un marco teórico contextual para el desarrollo de la presente tesis. Como fue mencionado anteriormente, en los capítulos siguientes plantearemos el modelo estructural del simulador desde el punto de vista del Análisis Orientado a Objetos, obteniendo los patrones de diseño que de él surjan y proponiendo la aplicación de otros existentes.

Actualmente sólo el AVS se encuentra bajo el modelo estructural por lo cual la propuesta aquí se hará con una visión más general tratando de abarcar todos los componentes del simulador.

Por último serán estudiados modelos aplicables a los subsistemas, siempre desde el punto de vista del Análisis Orientado a Objetos y la aplicación de Patrones de Diseño.

# Capítulo 5 Patrón AVS (Air Vehicle System)

*"Just because OO is wonderful does not mean that we have to throw away everything we learned in the 1970's and 1980's"* [YOU94]

Luego de haber presentado en la revisión teórica las diversas estructuras utilizadas en los simuladores de aeronaves, estos conceptos nos llevan al descubrimiento de un patrón de diseño que describiremos en este capítulo.

## 5.1 - AVS – Air Vehicle System (Estructural)

Se desprende de los modelos existentes que el uso de AVSM se encuentra generalizado y comprobada su eficacia, motivo por el cual vamos a extraer del mismo, y desarrollar desde el punto de vista de objetos un patrón de diseño de forma de establecerlo como un patrón estructural utilizado en los simuladores del dominio.

**Nombre:** AVS

**Intención:** Proveer una estructura flexible para la ejecución en tiempo real y sincronización de la simulación de una aeronave; permitiendo el cálculo y la comunicación externa e interna de los estados a través de los componentes que la forman, reproduciendo su estructura interna.

**Motivación:** Al simular una aeronave, es necesario utilizar un diseño que sea fácil de mantener. Este esquema simula a la aeronave real, imitando su diseño estructural, dividido en sistemas y subsistemas. Lo que permite mantener un grado mayor de integración a la realidad y subdividir un problema en unidades fácilmente comprensibles por personal que comprenda el problema físico real.

**Aplicabilidad:** Este modelo ha sido planteado y utilizado por el AVSM y DARTS.

**Estructura:**

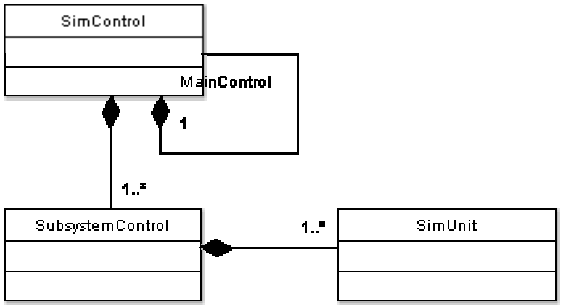


**Figura 21 AVS Structural Pattern**

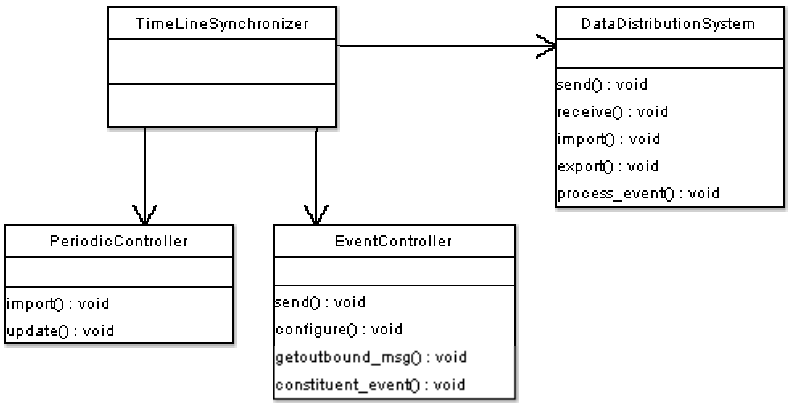


**Figura 22 SimControl: Esctructura Interna**

**Participantes:**

- SimControl: Es el responsable de las tareas de ejecución en tiempo real y sincronización de la simulación, además de la comunicación externa e interna de los estados.
- SubsystemControl: Se utiliza para servir como interfaz entre los componentes que encapsulan y el sistema, representando la simulación de un subsistema como un todo. Realizan el control y comunicación entre el SimControl y las SimUnits.
- SimUnit: Es el menor componente estructural que se encarga del cálculo de un estado dentro de la simulación.

**Colaboraciones:**

El SimControl comienza el intercambio sincronizado entre las áreas de exportación de datos de los SC (utilizando el método data_moves) del Data Distribution System (DDS), e invoca al PeriodicController (PC) y al

EventController(EC) en tiempos dados durante un frame. En una configuración multiproceso, un proceso asume la función de master TLS y corre en la mayor frecuencia que los otros procesos, utilizando comunicación interproceso (por ejemplo con semáforos) para coordinar el inicio de los frames, y períodos dentro del frame en todos los procesos.

El PC conduce todo el procesamiento periódico realizado por los subsistemas. Involucra invocar los subsistemas para realizar operaciones periódicas acorde a tiempos fijos. Es el encargado de llamar a los métodos periódicos de los SC. En la fase *Import* solicita que el PC invoque a los subsistemas para realizar su operación de import y luego *Update* que llama las correspondientes update de los subsistemas.

Luego el EC se utiliza para ejecutar el procesamiento aperiódico de los subsistemas, sirviendo de interfaz con la consola de instructor. Esta consola envía eventos a una cola indicando el comienzo o final de mal funcionamientos. El EC toma estos eventos como consecuencia a un llamado a *event_processing* y despacha el control al SC apropiado según una tabla de despacho acorde al evento procesado. Para su procesamiento el Ec requiere: determinar que SC recibe un evento, usando conocimiento del mapeo entre identificadores de eventos e instancias de subsistemas. Luego se debe invocar los subsistemas y extraer los datos requeridos de los eventos antes de la invocación.

El DDS provee la comunicación periódica externa. Cuando un evento es recibido, este lo pasa al EC para que lo despache al subsistema apropiado. Esto permite que el PC y el EC se mantengan ignorantes de los detalles de la estación de instructor y el modelo de ambiente. Todo se encapsula en el DDS y cualquier cambio no se propaga más allá de este.

**Consecuencia:**

La primera ventaja de la aplicación de Orientación a Objetos es que este comportamiento puede ser implementado por clases abstractas de las cuales deriven las que realizan el trabajo específico. Este patrón permite obtener una estructura definida del simulador, permitiendo concentrarse en el problema físico ya que modela en forma natural, muy cercana a la estructura real de la aeronave.

**Implementación:**

En el caso general, el AVS se implementa como multiproceso y multiprocesador. Mencionamos que la función del SimControl es unir los

subsistemas dentro de un proceso, mejorando la distribución de tiempo de sus subsistemas.

Los SimControl deben coordinar los procesos dentro del AVS, la comunicación externa al AVS de los estados y la sincronización general del sistema.

Según se había planteado y acorde a la bibliografía, tenemos dos cuestiones principales a encarar en el SimControl: *Distribución de datos y scheduling.*

**Usos Conocidos:** AVSM, DARTS, Simuladores SAAB.

**Patrones Relacionados:**

En este trabajo, hemos encontrado que el patrón de diseño AVS se encuentra emparentado especialmente con un patrón aplicable a sistemas de tiempo real denominado ***Recursive Control Pattern*** [SEL01] o ***Hierarchical Control Pattern*** [DOU02].

La simulación de vuelo puede ser categorizada dentro de los sistemas de tiempo real. En este tipo de sistemas, basado en el principio de Separation of Concerns y por razones de seguridad, es deseable mantener los aspectos de control lo más separadamente posible de los aspectos funcionales, es decir separar la interface de control de un objeto, de la interface funcional. En este patrón, el estado operacional de los componentes funcionales es controlado por un componente de control interno al sistema. Gráficamente:

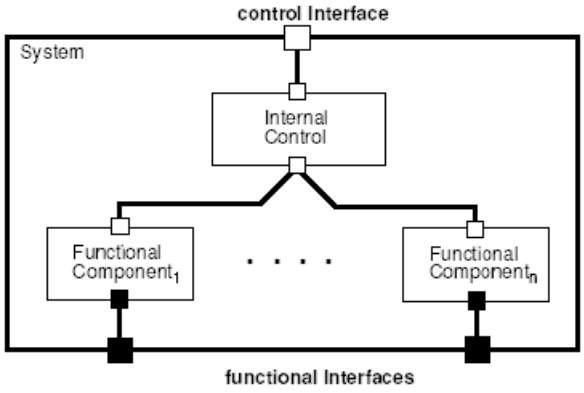


**Figura 23 Recursive Control Pattern**

Nótese que desde el punto de vista de los componentes funcionales, el elemento de control actúa como un controlador externo. Esta analogía sugiere una aplicación recursiva del patrón para componentes funcionales complejos que necesitan ser divididos en "subsistemas" más simples. Entonces la naturaleza recursiva del patrón indica que puede ser utilizada para sistemas arbitrariamente complejos.

Vemos que el principio de funcionamiento es prácticamente el mismo: cuando una entrada de control ingresa a través de la interface de control superior, el controlador interno traslada ésta a uno o más de los componentes funcionales de menor nivel. Si los componentes funcionales se descomponen más aún, estas señales son recibidas por los controladores internos, los cuales las trasladan a componentes funcionales de nivel inferior, y así sucesivamente. En el sentido inverso, los componentes funcionales, notifican a sus controladores de situaciones de error que requieran intervención del controlador. A su vez los componentes funcionales interactúan con clientes externos para proveer sus servicios respectivos.

La diferencia radica en que en AVS el controlador es, a la vez, comunicación con los clientes externos.

Los resultados son los mismos:

Puede definirse una clase abstracta de la cual desciendan los componentes, ahorrando tiempo de desarrollo y mantenimiento, unificando estrategias de control, generándose un sistema más confiable. Nótese que este tipo de herencia se puede hacer en la forma del patrón ***Template Method*** [GAM95].

Otro patrón relacionado es el ***Cyclic Executive Pattern*** [DOU02] el cual es utilizado sistemas cuya ejecución es altamente predecible (se puede determinar el punto de ejecución basado únicamente en el tiempo).

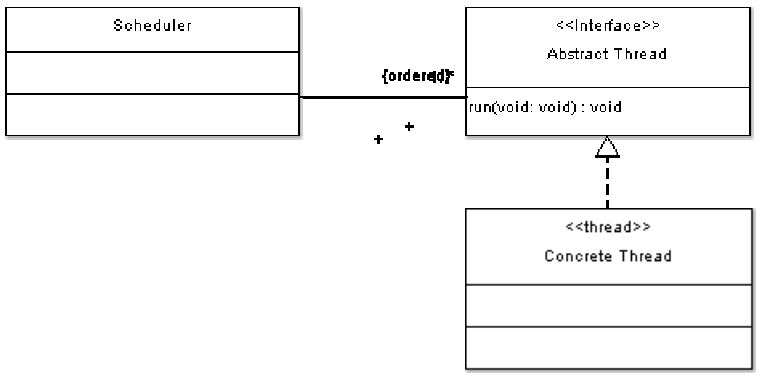

**Figura 24 Cyclic Executive Pattern**

Los elementos que lo componen son:

- *Abstract Thread:* Se utiliza como clase abstracta para concrete thread de forma de establecer una especificación / limitación de la estructura.
- *Concrete Thread:* Es un objeto activo construido principalmente para contener objetos pasivos que realizan el verdadero trabajo.
- *Scheduler:* Inicializa el sistema cargando todas las tareas y ejecutándolas cíclicamente. En general es responsabilidad de las tareas devolver el control de la CPU al scheduler cuando finalizan.

Este patrón en realidad es parte del AVS presentado anteriormente y puede utilizarse para implementar la relación PC con los SC y/o entre los SC y las SimUnit.

## 5.2 - Implementación

Debido a las exigencias en los cálculos involucrados en la simulación generalmente se utilizan múltiples procesadores o bien varios computadores en red para implementarla. El hardware utilizado generalmente es multiprocesador. Todos los SC se cargan en su procesador (elegido bajo algún criterio de balance de carga). Se instancia un SimControl para cada tarjeta de procesadores de forma de tener un SimControl para varios procesadores (misma tarjeta).

Se podrá optar por utilizar varios procesos o varios threads, donde la principal diferencia es el acceso a memoria. Si tenemos varios threads generados por el mismo proceso, estos comparten el mismo espacio de memoria, mientras que cada proceso posee el suyo propio. El hecho de estar en la misma área de memoria permite compartir datos entre threads utilizando por ejemplo punteros, mientras que los procesos necesitan mecanismos de IPC (Inter-process Communication) como memoria compartida (shared memory).

Cuando utilizamos modelos orientados a objetos y patrones de diseño, el problema a considerar se presenta con el uso del patrón ***Singleton*** [GAM95], por ejemplo cuando se requiere a un SC o el SimControl que devuelva una única instancia. En este caso, el acceso de varios threads implica el uso de mutex o semáforos o bien el uso de double buffering para realizar los cambios en las variables. Una tercer opción es separar la ejecución en bloques de eventos donde en un bloque los procesos actualizan los datos de los objetos que crearon y no se realiza IPC. En el próximo bloque los procesos se comunican utilizando métodos constantes de clases (este método es el preferido por introducir sincronización a nivel programa, implicando menor trabajo). La sincronización se asocia sólo con eventos: dentro de esos eventos cualquier acceso concurrente es seguro.

El uso de memoria compartida implica que un memory manager defina las posiciones de memoria donde se crearán los objetos, lo cual depende del OS y del compilador.

En el ambiente multi-thread es más simple ya que es el mismo espacio de memoria y por lo tanto todas las referencias a objetos son válidas y cualquier thread puede llamar un método virtual en cualquier objeto compartido que no haya creado él mismo. En este sentido el diseño multi-thread es más adecuado para orientación a objetos.

Para la implementación multi-thread querremos abstraer la implementación dependiente del sistema operativo para generalizarla. Para esto utilizaremos un patrón ***bridge*** [GAM95] junto con un patrón ***abstract factory*** [GAM95], lo cual implica una separación de la interface con la implementación concreta.

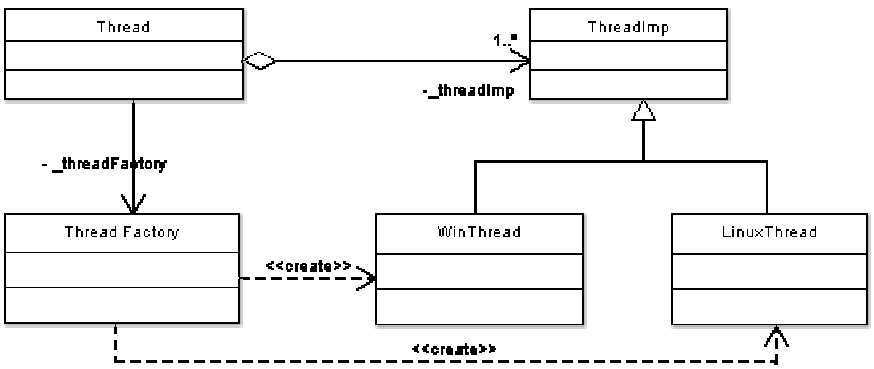


**Figura 25 Abstracción de Threads**

De esta forma inclusive tendremos que cada objeto activo descenderá de la clase Thread, la cual esconde la implementación dependiente del sistema operativo.

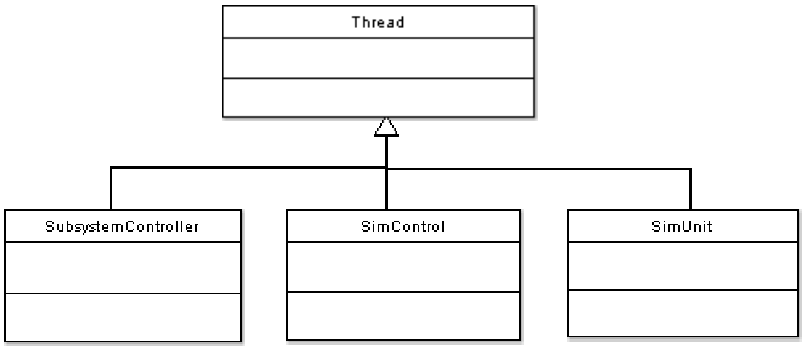


**Figura 26 Jerarquía de Threads**

# Capítulo 6 Distribución de Datos (DDS)

Se ha presentado en el capítulo anterior el patrón de diseño que define la estructura del simulador, pero necesitamos ahora el desarrollo de algunos modelos para la implementación del mismo. La aeronave posee numerosos sistemas relacionados (físicos y ficticios) que deben comunicarse entre si, motivo por el cual la transmisión de estos datos es de vital importancia.

## 6.1 - Datos y su organización

Es inevitable tener que compartir variables entre subsistemas (por ejemplo el valor del ángulo de ataque; la velocidad; altitud; etc.) ya que las mismas variables pueden ser utilizadas en diferentes subsistemas para realizar sus cálculos o bien en la representación en las estaciones gráficas.

En todos los planteos [SJO02], el sistema de distribución de datos (DDS) es parte del SimControl, tratando de ayudar a la abstracción del hardware de forma de ser utilizado en diferentes simuladores con mínimos cambios. Las soluciones generalmente utilizan memoria compartida con o sin double buffering para implementarlo cuya principal ventaja es la velocidad, pero el problema con esto es que alguien puede destruir datos necesarios para otros subsistemas. Una solución es definir el orden de ejecución (TLS) pero implica mantener esta secuencia, lo que no siempre es fácil cuando se presentan cambios en los componentes.

La variante con double buffering es una solución inclusive en el caso de múltiples procesadores, ya que lo que asegura es que varios subsistemas recibirán los mismos valores para una variable sin importar el orden de ejecución. En este caso los valores se obtienen de un buffer generado en el ciclo anterior y se escriben a otro buffer que será intercambiado al finalizar el ciclo actual.

Resumiendo, el principal objeto de estudio cuando se comparten variables en sistemas distribuidos en tiempo real es garantizar los correctos valores de los mismos en todo momento.

## 6.2 - Modelo de Datos

Para lograr la integridad y consistencia de datos entre subsistemas, los mismos se comunicarán utilizando el patrón ***Communicator*** [CRI93], con lo cual los subsistemas no se comunicarán directamente sino que lo harán a través de su SimControl. A nivel subsistema sucederá lo mismo: los componentes no se comunicarán directamente sino que lo harán a través de su SC.

Entonces, en general tenemos un conjunto de subsistemas comunicados entre si, comunicación que puede llevarse a cabo por múltiples medios como memoria compartida, redes ethernet, etc., e inclusive su implementación variar según se requiera utilizar double buffering u otro método. Para mejorar el modelo de distribución de datos utilizamos un ***Distributed Proxy Pattern*** [SIL97] utilizando el esquema de datos provisto por el ***Data Bus Pattern*** [DOU02] el cual provee un bus común (lógico) en el cual múltiples servers transmiten información y múltiples clientes pueden recibir datos y eventos. Básicamente es un ***Proxy Pattern*** [GAM95] con almacenamiento centralizado en el cual se pueden conectar varios objetos de datos.

La versión que utilizaremos es la denominada versión de Recepción (PULL), dónde los clientes chequean el bus para obtener nuevos datos y es el mostrado en la siguiente figura:

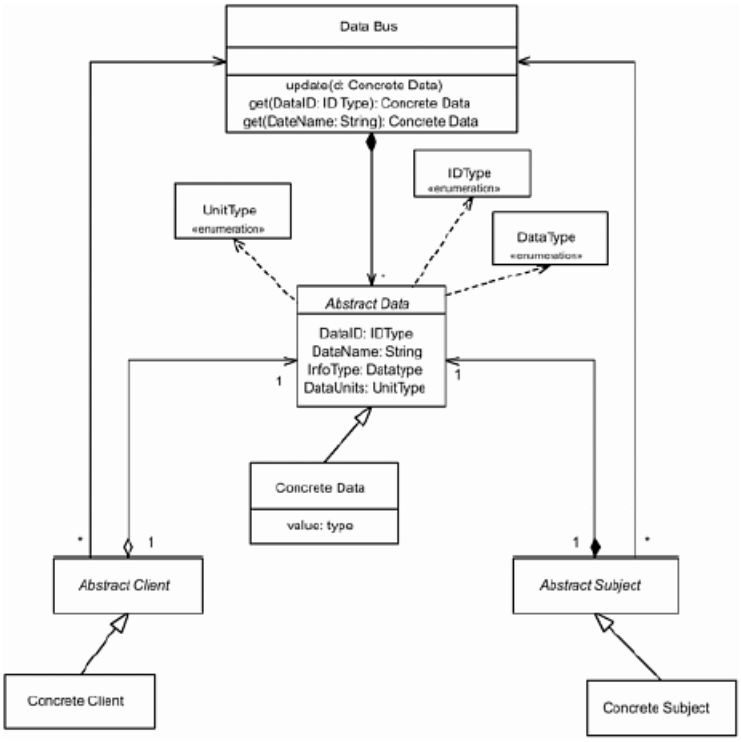


**Figura 27 Data Bus Pattern (PULL)**

Los elementos que intervienen en ambos son:

- *Abstract Data:* Especifica la estructura de todos los objetos de datos que pueden ser conectados al bus. Los atributos son:
  - *ID Type:* Podría ser una enumeración como {Velocidad del Aire=0, Alpha=1, Tita=2, Temperatura Ambiente=3, Altitud=4}
  - *Data Name:* Nombre del dato como alternativa a su búsqueda.
  - *Data Units:* Otra enumeración identificando el tipo de dato almacenado.
- *Abstract Client:* En (PULL) solicita datos cuando desea chequear por nuevos datos. En (PUSH) es notificado por el Listener cuando hay nuevos datos. El listener especifica el Data ID al cual desea subscribirse.
- *Abstract Subject:* Es el proveedor de datos para el bus.
- *Concrete Data:* Es la subclase de datos específica y única para todo el sistema.

- *Concrete Client:* Es la subclase del Abstract Client que utiliza los objetos de datos obtenidos de los Abstract Subjects a través del bus de datos.
- *Concrete Subject:* Instanciable subclase de Abstract Subject.
- *Data Bus:* Provee la ubicación centralizada para los objetos de datos a ser compartidos.

Mencionamos el uso del patrón ***Distributed Proxy*** [SIL97] de forma de desacoplar la comunicación de la funcionalidad específica del objeto. En este patrón los objetos que intervienen son:

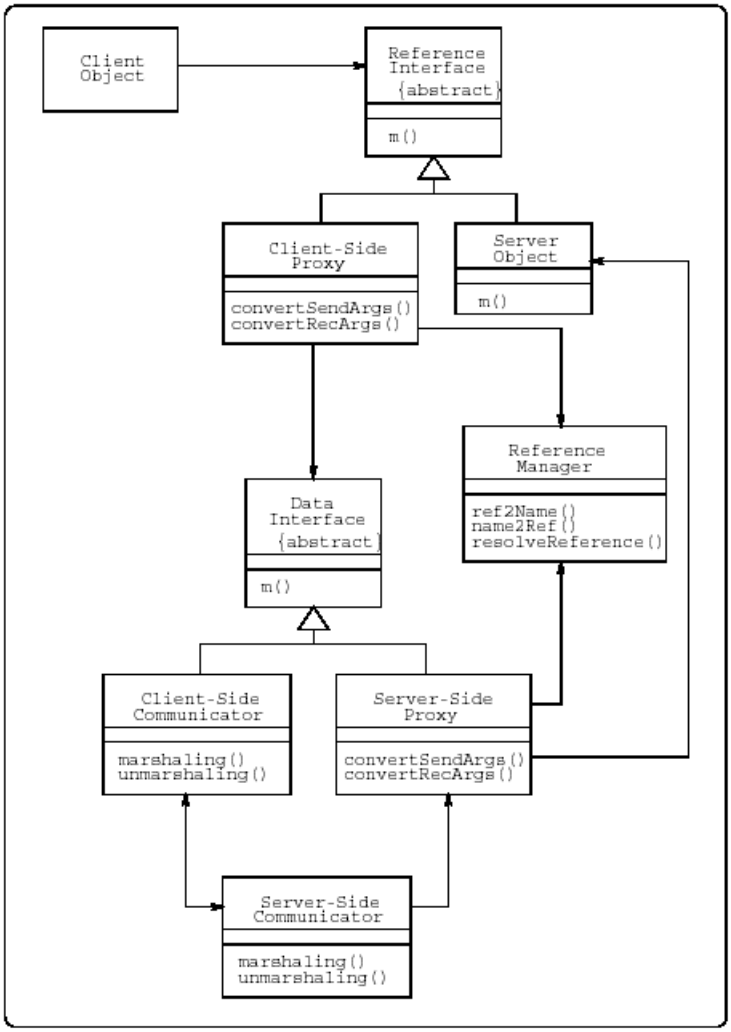


**Figura 28 DProxy Design Pattern**

- Client Object: Requiere un servicio del Server Object (invoca uno de sus métodos)

- Server Object: Provee de servicios al Client Object
- Client-Side Proxy: Representa al Server Object en el nodo del cliente. Utiliza al Reference manager para convertir las referencias a objetos a nombres distribuidos y viceversa.
- Client-Side Proxy: Es el punto de entrada de las solicitudes remotas al Server Object.
- Referente Manager: Es el responsable de asociar referencias a objetos con nombres distribuidos y viceversa.
- Distributed Name: Es un identificador válido para todos los nodos.
- Client-Side Communicator y Server-Side Communicator: Realizan la comunicación física. Poseen métodos para convertir lamadas a métodos a arreglos de bytes.
- Reference Interface: Define la interface para el Server Object y el Client Side proxy.
- Name Interface: Define una interface común al Server-Side y Server-Side Communicator.

La comunicación de datos entre SimControls puede realizarse utilizando diferentes opciones de hardware dependiendo de la implementación en que nos encontremos, pero en el caso más general supongamos diferentes computadores conectados entre sí utilizando por ejemplo una red ethernet. Obviamente se debe tratar de minimizar la comunicación entre SimControls, lo que se hace con una estratégica distribución de los subsistemas, de forma que los que posean mucha comunicación de datos queden bajo la órbita del mismo SimControl. El modelo puede ser reducido luego y reutilizado en el caso de encontrarnos con otros tipos de configuraciones de hardware e incluso sería el caso de la aplicación en la comunicación de datos entre SC's.

En la figura siguiente se muestra el modelo propuesto para el sistema de distribución de datos. El SimControl posee una referencia al DDS el cual es visto como un objeto local pero en realidad es un proxy que encapsula la comunicación con el verdadero servidor del bus de datos (DataBus) que puede ser remoto o local. El SimControl utiliza los métodos de su DDS para enviar y recibir datos, los cuales están identificados unívocamente en todo el simulador por sus código de identificación (alfa, beta, TAS, etc.). De esta forma el SimControl solicitará del DDS todas las variables que utilizan sus

subsistemas, las cuales serán transferidas a los SC. Luego los SC serán ejecutados y recibirá las variables producidas por éstos para ser transferidas al DDS y que puedan ser utilizadas por otros SimControl.

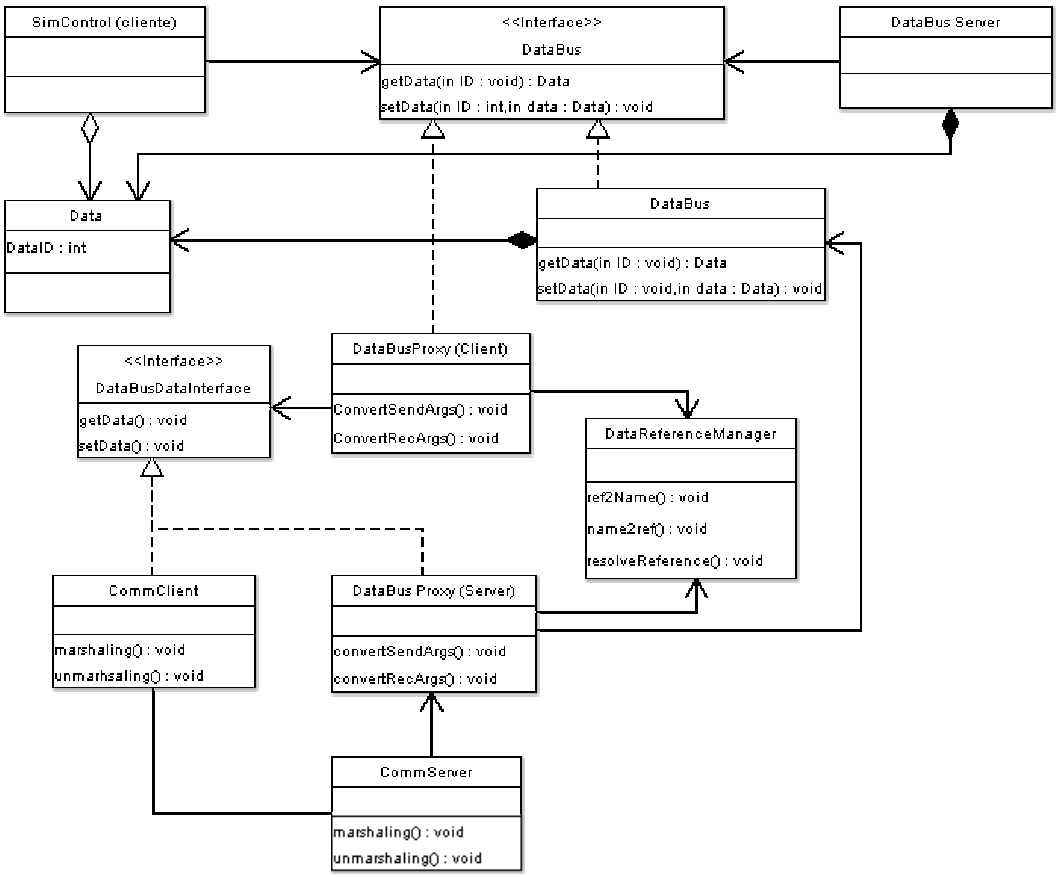


**Figura 29 Modelo del DDS**

A nivel de comunicación entre subsistemas, tendremos que el SimControl controla una serie de subsistemas. Para evitar la dependencia los datos que consuma el subsistema, así como los que produzca, serán registrados por el SC durante la etapa de inicialización guardando las referencias a dichas relaciones. El modelo a nivel subsistema será un caso particular del anterior, ya que generalmente los datos se operan localmente a nivel hardware, pudiéndose reducir el modelo según se muestra en la siguiente figura.

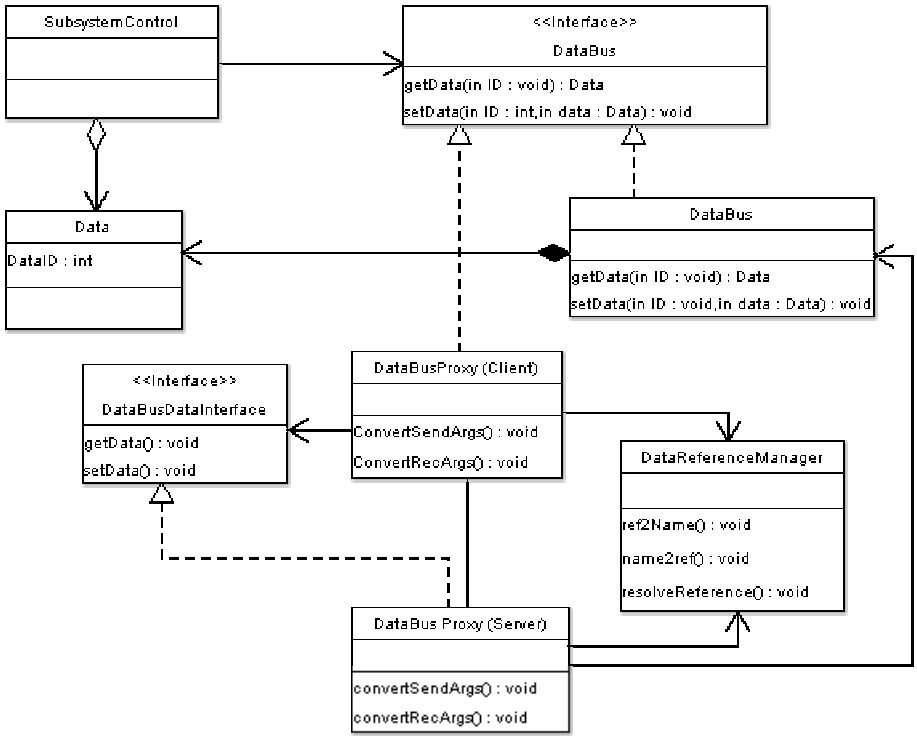


**Figura 30 Modelo del DDS a nivel Subsistema**

El SimControl llamará a los subsistemas a una primer fase en la que se producirá el paso de los datos consumidos a cada SC. Estos los almacenarán en sus DDS. Luego se producirá la fase de ejecución de los SU gobernada por el SC y al finalizar el SC toma los datos producidos para ser pasados al SimControl y que sean consumidos por otros SC. La ejecución de los SU podrá ser en paralelo o en secuencia acorde a las necesidades del caso. Es decir para asegurar la integridad y consistencia de datos los SU no intercambian datos con el DDS sino que lo hacen a través de su SC.

Del análisis anterior vemos que existirán tres métodos relacionados con esto en los SC:

1. INIT: Se llamará al inicio de la simulación para realizar la inicialización de tablas y DDS local.
2. RETRIEVEINFORMATION: Este será llamado inicialmente para conocer las variables necesarias para ejecutar este subsistema y las que producirá el mismo. También parámetros de ejecución como su frecuencia de actualización.
3. EXECUTE: El método llamado por el SimControl para que el SC realice sus funciones y retorne los valores actualizados de las variables que produce.

## 6.3 - Distribución de Datos – Sistemas Mixtos

Luego de presentar el modelo anterior, es preciso analizar el caso en que se deben integrar dispositivos reales junto a los simulados. Los buses de datos en aeronáutica siguen básicamente dos especificaciones: ARINC 429 para aeronaves comerciales y de transporte (Airbus 310/320/330/340; Helicópteros Bell; Boeing 727/737/747/757/767; McDonnel Douglas MD11; etc.), y el MIL-STD-1553 para el caso de aeronaves de uso militar.

En este caso cada dispositivo (subsistema) se alimenta de una determinada cantidad de parámetros provenientes de otros dispositivos los cuales podrán ser reales o simulados. Para este caso el modelo es el mismo, la única diferencia es que los SC encapsulan a estos dispositivos reales mediante el uso del patrón de diseño ***Bridge*** [GAM95].

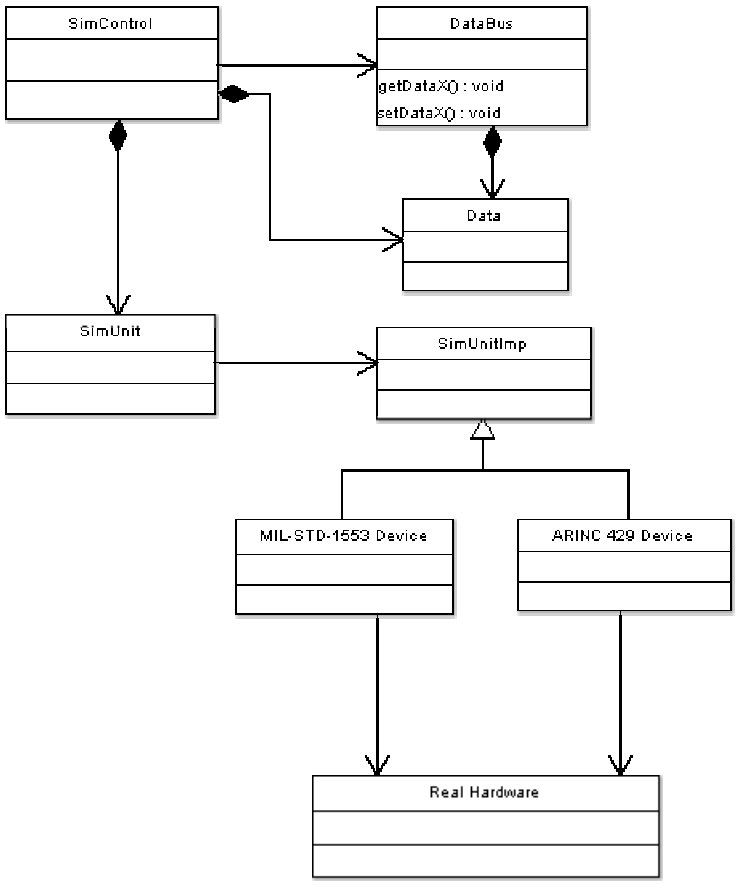


**Figura 31 Subsistema Mixto**

De esta forma el dispositivo real o simulado queda encapsulado y es transparente a la operación del simulador. Sobre este tema volveremos en el tratamiento de la cabina de vuelo.

# Capítulo 7 Subsistemas

Hasta el momento vimos que la simulación de una aeronave se realiza representando cada uno de los subsistemas que la componen, estableciendo una estructura de objetos que interactúan para reproducir su comportamiento. Para ir desarrollando en profundidad cada parte componente de la estructura, en este capítulo, debemos analizar específicamente lo que sucede con los subsistemas que componen una aeronave y que en consecuencia se verán simulados dentro de nuestro modelo.

## 7.1 - Modelos

En general los subsistemas representados serán:

1. **Aerodinámico:** Se encontrará formado por los componentes relativos al cálculo de los aspectos aerodinámicos que gobiernan la aeronave como momentos por fuerzas aerodinámicas; coeficientes de sustentación de ala y empenajes, etc.
2. **Alerta y precaución:** agruparemos aquí los componentes relativos a los sistemas de alerta de la aeronave como situación de pérdida; incendio; etc.
3. **Control:** componentes relativos a la estabilidad y control de la aeronave. Aquí tendremos los componentes de cálculo de estabilidad longitudinal y lateral de la aeronave.
4. **Combustible:** Relacionado con el subsistema de propulsión, representará el sistema de combustible permitiendo definir consumos, capacidad, etc.

5. **Hidráulico:** Representación del sistema homónimo en la aeronave que se encuentra relacionado al accionamiento de superficies de control; tren de aterrizaje; etc.
6. **Tren de Aterrizaje:** agrupará los componentes relativos al tren y tendrá relación con el subsistema hidráulico; alerta y precaución; afectando al aerodinámico y por lo tanto al control.
7. **Navegación:** componentes relativos a los sistemas VOR; ILS; Radio comunicaciones; radar meteorológico; etc.
8. **Propulsión:** componentes que se encargan del cálculo de estado de la planta de poder de la aeronave y que se encuetran relacionados con el sistema de combustible e hidráulico.
9. **Armas:** componentes específicos a este sistema.

Por lo tanto de lo planteado hasta el momento tendremos que la estructura de la aeronave sería:

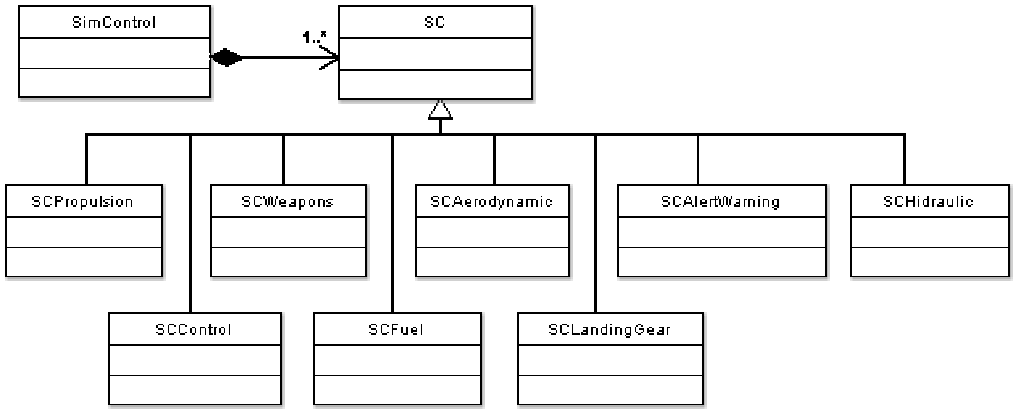


**Figura 32 Subsistemas**

Desarrollando un poco más este modelo podemos tomar el subsistema aerodinámico dónde entre los estados necesarios de cálculo tendremos la representación del ala y sus parámetros. Tendremos entonces representados los componentes principales que afectan a los cálculos de la aerodinámica de la aeronave según se presenta en la figura:

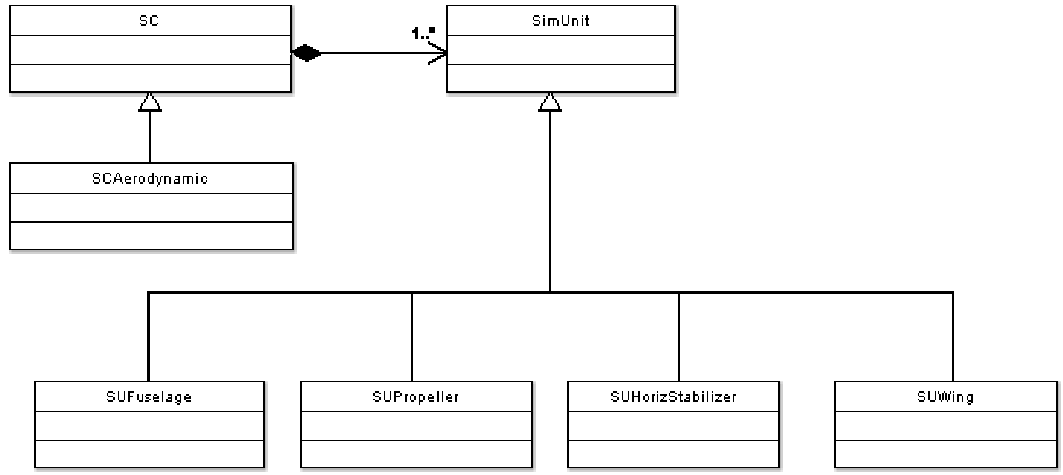


**Figura 33 Subsistema Aerodinámico**

Desarrollamos ahora en especial lo relativo al cálculo del CLw y Cmw de la misma, donde planteamos:

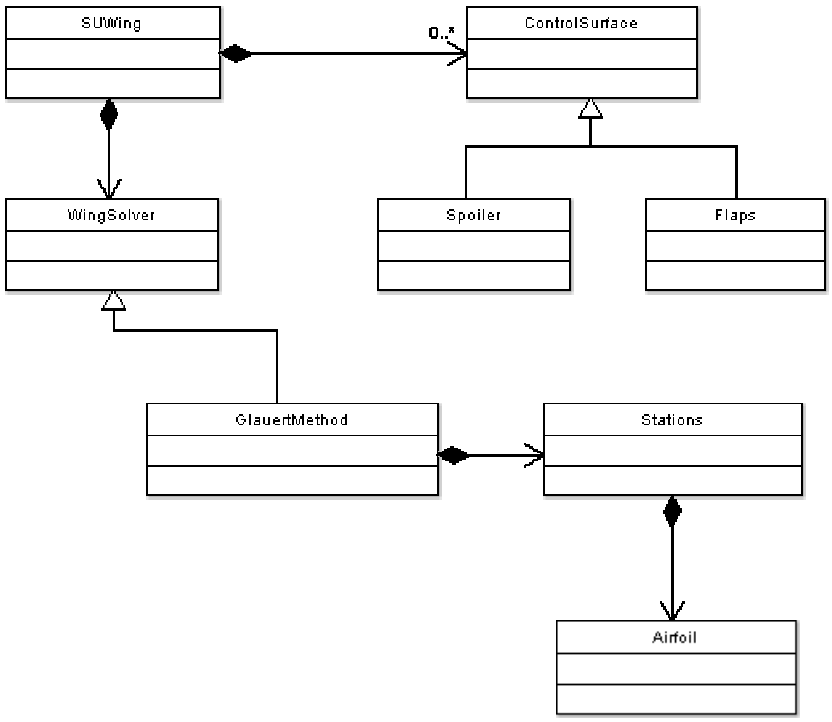


**Figura 34 Subsistema Aerodinámico - Glauert**

En este caso se ha representado la parte correspondiente al ala pero sería lo mismo para el estabilizador horizontal. Poseemos el ala la cual como propiedades posee un Cl y un Cm tridimensionales que pueden ser calculados utilizando la clase WingSolver. El método utilizado puede variar y en nuestro

caso planteamos el uso del método de Glauert como una de las posibilidades gracias al uso del patrón ***Strategy*** [GAM95]. En la figura también se muestra que este método se vale de dividir el ala en estaciones de las cuales uno de los parámetros de cálculo es el perfil alar, representado aquí por la clase Airfoil.

El uso del patrón Stragegy nos permite trabajar con diferentes algoritmos de cálculo y poder analizar los resultados, mientas el resto de la simulación no se ve afectado.

Al analizar la relación que existe en los subsistemas (Subsystem controller – Componentes) vemos que los subsistemas encapsulan una gran cantidad de componentes con los cuales debe comunicarse la simulación. El problema que se presenta es el intercambio de datos, que es el mismo problema tratado en el patrón ***Communicator*** [CRI93] o el ***Mediator*** [GAM95].

Por ejemplo:

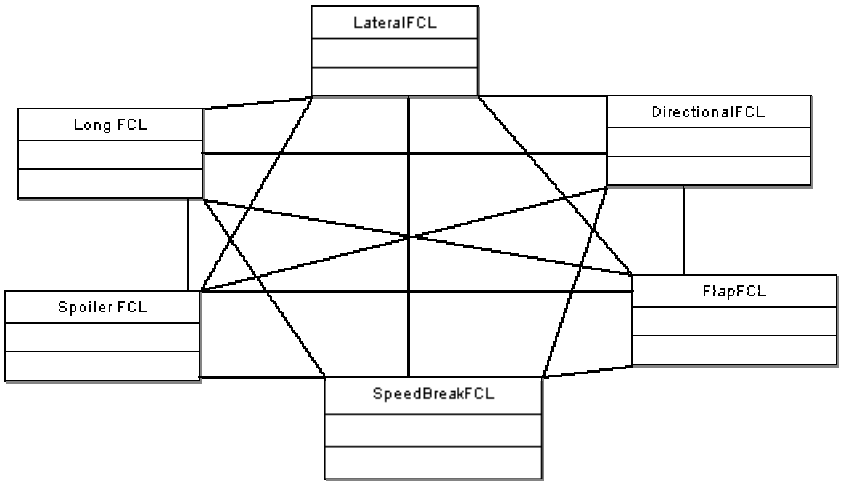


**Figura 35 Problema de Comunicación**

En la figura tenemos diferentes componentes que debe comunicarse entre sí (FCL=Control de vuelo longitudinal, nuestro subsistema). Cada clase depende de las demás de las cuales necesita datos, esto crea un sistema acoplado donde el comportamiento agregado a una clase afecta a todas las demás, lo cual tiende a ser el comportamiento de una única clase que limita el reuso, testeo y mantenimiento.

En este tipo de sistemas tenemos que se requieren n*(n-1) mensajes además de redundancia en el diseño de almacenamiento y computación. El caso general es una dependencia nxn que requiere n*(n-1) mensajes para la comunicación de estado de los componentes.

Aplicando el patrón ***Communicator*** [CRI93] tendremos:

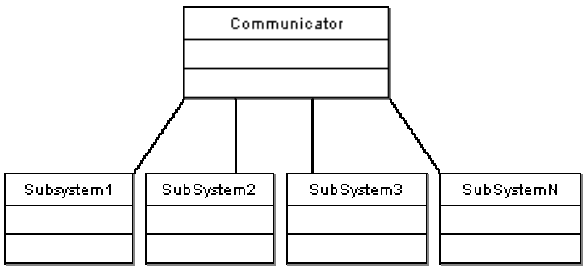


**Figura 36 Communicator Pattern**

En la figura, cuando un componente necesita comunicar su estado lo envía al communicator (SC), el cual colecta esa información. Acto seguido el SC envía los datos necesarios a cada componente. Existen dos formas posibles: cada componente envía su estado cuando quiera y el SC espera o bien, él controla la ejecución, lo cual reduce el tiempo de cómputo.

Otros casos dónde se presenta el mismo problema es el ambiente, ya que cada agente en el ambiente debe conocer objetos en sus cercanías (combate aéreo, tráfico cercano, aeropuertos, etc.).

El pattern reduce a 2xn mensajes disminuyendo el tráfico en la red. El overhead en la comunicación resulta una ventaja de todas formas debido a la disminución de mensajes. El problema de la caída de la red puede tratarse con redundancia.

La principal diferencia con el patrón ***Mediator*** [GAM95] es que en communicator existe una relación nxn contra del Mediator 1xn. El propósito de utilizar el patrón ***Mediator*** [GAM95] es que los objetos componentes del subsistema deben colaborar entre ellos pero no conocerse (no interactuar directamente), y es por eso que lo hacen con un tercero que los mantiene coordinados. Por lo tanto la estructura del subsistema nos queda:

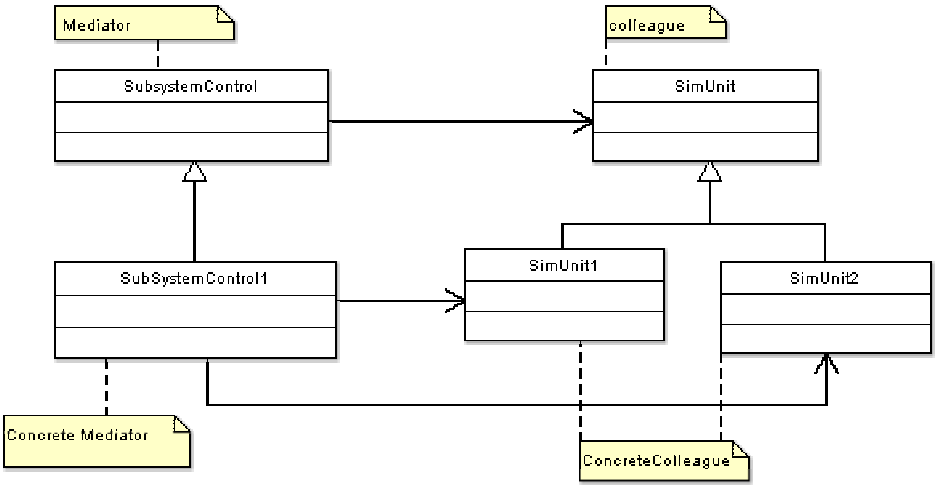


**Figura 37 Subsystem Pattern**

Analizando LaRS++ [CUN99], tenemos que no existen subsistemas pero existen sistemas que componen el vehículo, los cuales se encuentran implementados siguiendo la estructura del patrón ***Mediator*** [GAM95], con el cual vemos que la solución concuerda con la estructura del AVSM y de LaRS++. El beneficio directo de utilizar el patrón ***Mediator*** [GAM95] es el incremento en la testeabilidad y reuso.

## 7.2 - Abstracción del Hardware

Los subsistemas necesitan comunicarse con el hardware y acorde a las necesidades que persiga la simulación, tenemos simuladores que reproducen los sistemas de la aeronave real, totalmente por software; otros que controlan los sistemas de una aeronave real o bien aquellos que podríamos considerar mixtos e integran sistemas reales con sistemas simulados por software. Además del hardware propio de la aeronave, dentro de la amplitud de hardware utilizado, una de las interfaces más utilizadas son placas adquisidoras de datos las cuales presentan características propias de cada fabricante en cuanto a su utilización. Estas placas poseen librerías para utilizarlas, las cuales dependen del fabricante, provocando una dependencia muy grande del software de simulación de la configuración de hardware con que se cuente.

Uno de los problemas que se presentan en el diseño de la simulación es cómo interactuar con el hardware. Si el modelo se plantea demasiado dependiente del hardware, se volverá difícil de comprender, modificar y

analizar problemas. El diseño buscado debe acoplar clases comunes por herencia mientras que desacople las que no lo están. El modelo está formado por dos capas principalmente: La capa de servicio que brinda la funcionalidad (Interface) que representa al dispositivo en si mismo, mientras que la capa del driver realiza la función específica.

No debe olvidarse que además puede desearse utilizar elementos simulados por software para determinadas situaciones de investigación y cambiarlos por elementos reales para otros estudios.

Los drivers son un ***Bridge*** [GAM95] que desacopla abstracción de implementación y se utiliza cuando la implementación debe ser escondida del cliente además de elegirse en tiempo de ejecución. El servicio abstrae lo relativo a comunicaciones con el driver y es una variación del ***Mediator*** [GAM95]. La relación driver-servicio se construye con el patrón ***Builder*** [GAM95] pero podría utilizarse cualquier patrón creacional.

De aquí que al generalizar tendremos definidas interfaces que representen los instrumentos y comandos de la aeronave: flaps, timón de dirección; pedales; etc. Mediante estas interfaces quedará definida la forma de enviar y recibir datos (comunicación con el instrumento), mientras que la implementación de estas interfaces dependerá del hardware asociado (por ejemplo un IO de la placa adquisidora de datos, un mouse, etc.).

En general tendremos:

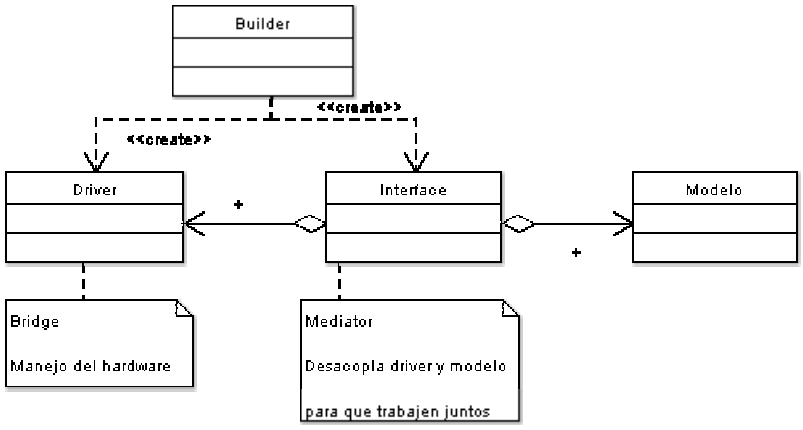


**Figura 38 Abstracción del Hardware**

Tomando la estructura del gráfico, veamos el caso del hardware para comandar el alerón como un caso en particular:

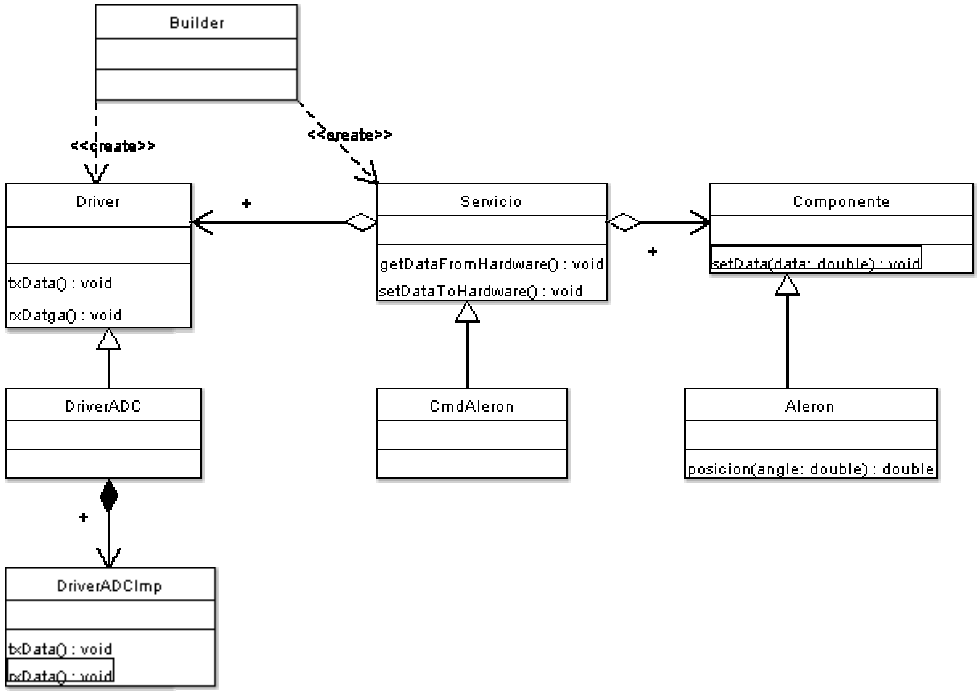


**Figura 39 Abstracción hardware de Alerón**

Entonces tendremos drivers para comunicarnos de la siguiente forma (proceso cíclico):

- De la información del hardware actual son creados los drivers por el builder.
- Se construye la lista de drivers (hardware).
- Se construye la lista de servicios.
- A cada servicio se le asigna una referencia al modelo y al driver.
- En tiempo de ejecución se obtienen los datos desde los drivers.
- Luego los servicios pasan los datos al modelo
- Se ejecutan los cálculos necesarios
- Los servicios pasan datos al driver y este al hardware.

Nótese que esto puede ser replicado al nivel deseado. O sea, utilizando reiteradamente el patrón bridge se pueden establecer tantas capas o niveles de abstracción como sean requeridos.

## 7.3 - Otros patrones

Cuando se están realizando estudios de parámetros de la aeronave, lo que se requiere es un análisis de determinadas variables para ver sus valores promedios, máximos o mínimos, por lo cual es útil el uso del ***patrón Tally*** [SIMM94]. Para esto se crea un Tally asociado a la variable que monitorea sus cambios, exponiendo la siguiente interface:

- ***Create*** para inicialización
- ***Observation*** llamada cada vez que ocurre un cambio en la variable
- ***Result*** para computar y devolver el resultado

Es decir que en el método Observation se implementa la operación deseada.

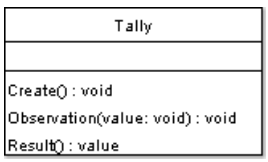


**Figura 40 Tally Design Pattern**

# Capítulo 8 Interface Gráfica

Lo que hemos presentado hasta el momento constituye un modelo completo de la aeronave como sistema, pero sin considerar ningún tipo de interacción con el exterior (piloto, instructor, etc.), y por consiguiente sin posibilidad de control humano. En este capítulo plantearemos los modelos que simulan las interfaces con el humano y la forma de control sobre la aeronave.

## 8.1 - Planteo General (Cabina de Vuelo)

Existen varios sistemas visuales que intervienen en la simulación: el ambiente y los distintos instrumentos que forman la cabina. La parte visible del ambiente no requiere interacción humana sino que es simplemente la representación gráfica del mundo. Los instrumentos representan parámetros y en algunos casos deben recibir las acciones del usuario que afectarán a otros subsistemas (a la simulación en si misma).

La GUI no debe interferir con la ejecución de la simulación para lo cual se necesita que se encuentre completamente aislada del modelo. Por otra parte se va a desear el cambio de GUI: esto es, se necesita optar por diferentes tipos de instrumentos simulando analógicos, digitales, aeronaves en sus diferentes configuraciones, mezcla con hardware real, etc. Por suerte, existe una tendencia en las aeronaves actuales de representar los instrumentos tradicionales en pantallas LCD y esto facilita notablemente que el simulador pueda adaptarse dinámicamente a las diferentes configuraciones de aeronaves. Toto esto implica un problema ya que se van a tener redes heterogéneas para comunicarse con las GUI y es por esto que se necesita desacoplar lo máximo posible la simulación de la GUI al punto que si hay problemas de comunicación con la GUI, esto no debería “congelar” la simulación.

Para solucionar esto se plantea el uso de una combinación del patrón ***Tokenizer*** [EKS01] y del ***Facade*** [GAM95]: entonces la cabina puede ser una unidad funcional (facade) que utilice el tokenizer para comunicarse con el resto de la simulación.

La cabina de vuelo será presentada como una unidad funcional implementada utilizando el patrón ***Facade*** [GAM95] el cual presenta una interface unificada a un conjunto de interfaces en un subsistema. Con esto el subsistema de la cabina de vuelo queda definido por una clase de alto nivel que lo hace más fácil de utilizar. La figura nos muestra la estructura de este patrón:

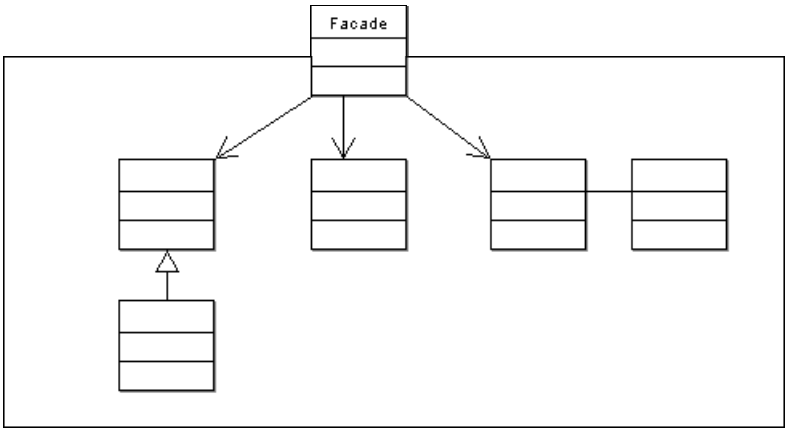


**Figura 41 Facade Design Pattern**

A su vez, de lo dicho anteriormente, la comunicación con la simulación debe ser independiente de la red y sin interferencias, para lo cual utilizamos el patrón ***Tokenizer***. En la figura se muestran los componentes que intervienen en el patrón ***Tokenizer***:

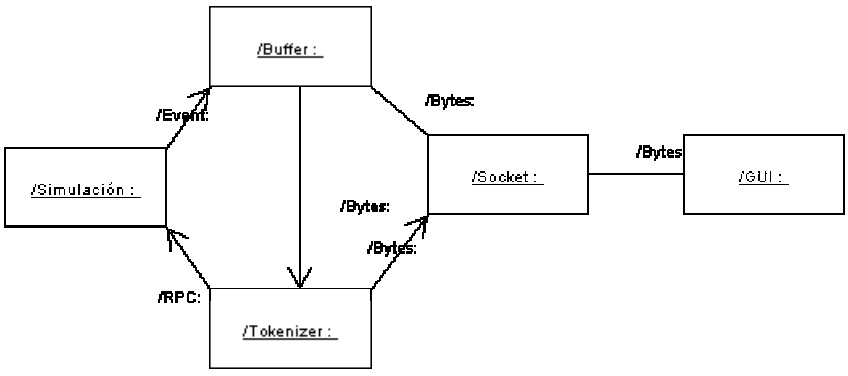


**Figura 42 Tokenizer Design Pattern**

El patrón tokenizer provee una interface independiente del lenguaje de programación, la cual tiene mínima interferencia con la simulación. La comunicación se lleva a cabo, cuando se envía una representación del comando al socket. El proceso buffer lee del socket y lo envía al proceso tokenizer, el cual lo convierte (la secuencia de bytes) en algo comprensible

para el lenguaje de implementación de la simulación o de la GUI (acorde al sentido de la comunicación). Cuando es necesario actualizar la GUI, el tokenizer envía el comando respondiendo a un mecanismo de subscripción implementado con el patrón ***Observer*** [GAM95].

Como la cabina de esta forma es un objeto distribuido, procederemos a aplicar el patrón ***Distributed Proxy*** [SIL97] utilizado anteriormente para el DDS, de forma de desacoplar la comunicación de la funcionalidad específica del objeto.

Por lo tanto, modelando la cabina de vuelo acorde a lo planteado hasta el momento tendremos:

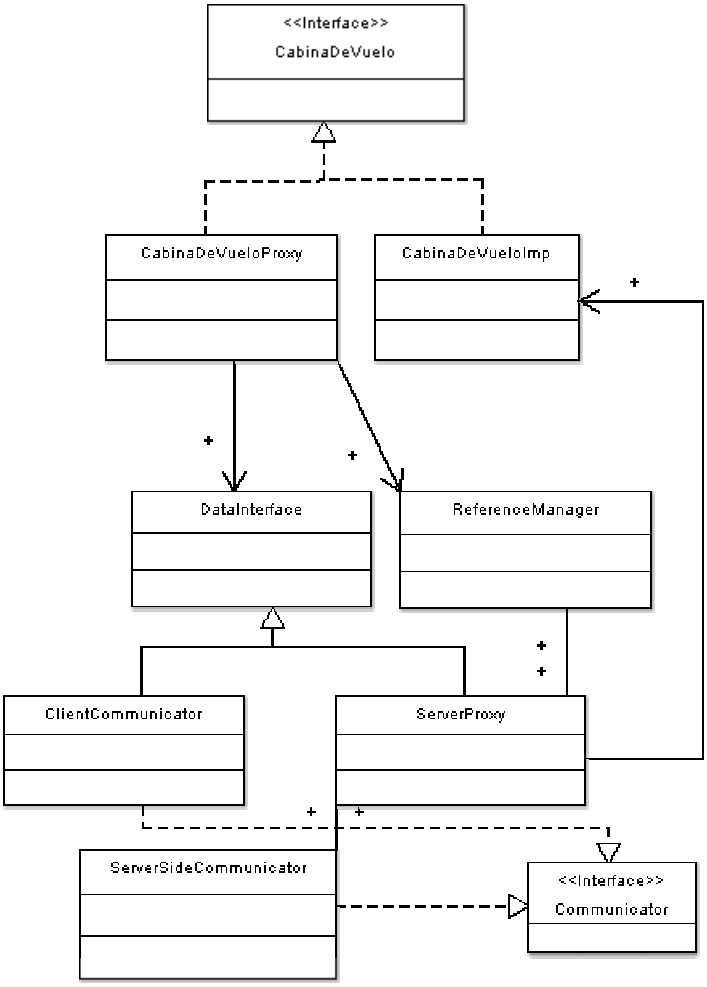


**Figura 43 Cabina de Vuelo**

Generalizando, la solución adoptada con el patrón tokenizer se aplica a las demás consolas que intervienen en la simulación de las cuales pueden implementarse acorde a las necesidades para visualizar los distintos tipos de

parámetros dentro de la simulación, flexibilidad lograda gracias al uso del patrón MVC. Así podremos tener consolas que muestre especificacmente parámetros de navegación; visualización de parámetros de sistemas bajo estudio; la consola del entrenador; etc. Lo que es importante de notar es que si la cabina de vuelo como unidad funcional se encontrara ligada al resto de la simulación por otro medio que no sea una red ethernet (por ejemplo VME) esta encapsulación haría transparente la nueva implementación.

Pasaremos ahora a plantear el modelo interno de la cabina de vuelo.

## 8.2 - Instumentos Simulados

Las cabinas modernas utilizan instrumentos que se presentan en monitores, que pueden ser representados sin problemas en monitores de PC por la simulación, ya que estos instrumentos modernos en general están representando instrumentos analógicos tradicionales. Estos instrumentos tradicionales podemos agruparlos en dos grandes grupos como: instrumentos con cuadrantes tipo relojería o instrumentos de banda. Existen casos específicos especiales como el FDI y el HSI que poseen características propias, como las V bars o el horizonte artificial. Analizando los instrumentos, vemos que existe una estructura de elementos más simples que los forman como marcas en forma de triángulo, o bien dos instrumentos de banda que forman uno solo. Por ejemplo, en el caso de indicación de velocidad podemos tener machímetro y velocímetro en el mismo instrumento representados por dos indicadores de tipo de banda.

La siguiente figura muestra un FDI analógico con los elementos que lo componen:

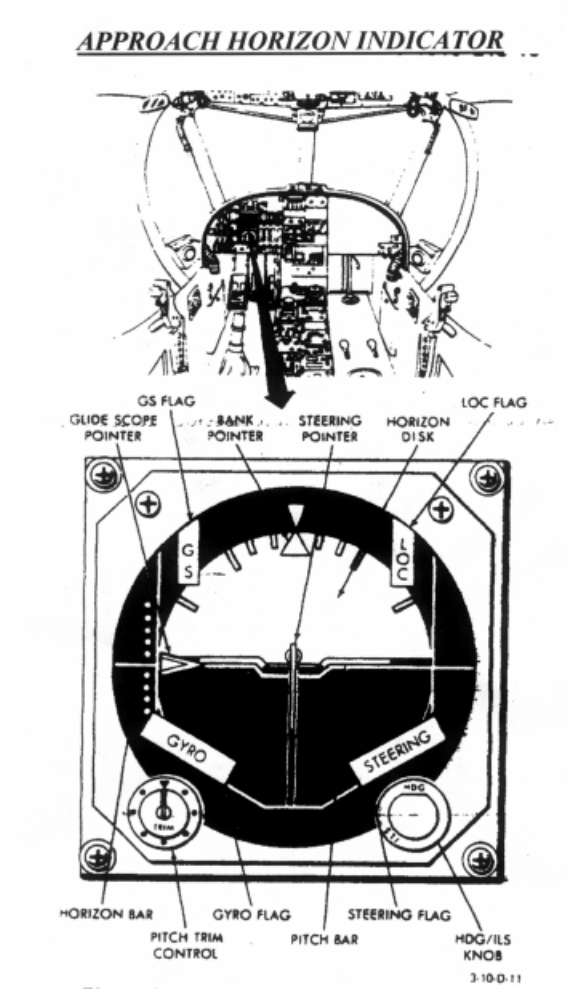


**Figura 44 FDI Analógico**

En la siguiente figura se muestra un HCI analógico con sus elementos constitutivos:

**Figura 45 HCI Analógico**

A continuación se pueden ver los instrumentos anteriores pero en su presentación digital:


**Figura 46 HCI y FDI en presentación digital**

El MVC posee una organización estructural adecuada para sistemas interactivos, dentro de los cuales, obviamente se encuentra la cabina de vuelo. Este modelo divide a la aplicación en:

- *Model*: se encarga de la funcionalidad y los datos.
- *View:* despliega la información al usuario
- *Controller:* maneja la entrada del usuario

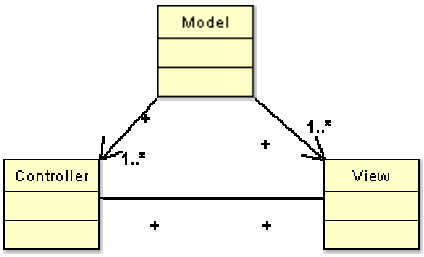


**Figura 47 MVC**

Si bien los instrumentos muestran la interface gráfica, sus datos y funcionalidad estan distribuidos en otros objetos. Esto surge de aplicar el patrón arquitectónico MVC ya que el modelo implica la funcionalidad y los datos, la vista que da la información al usuario y el controlador que maneja la entrada del usuario. Estas entradas son convertidas en requerimientos a la vista o al modelo.

Para mejorar la construcción de instrumentos de cabina proponemos el uso del patrón ***composite [GAM95]*** ya que presenta la mejor forma de organizar estructuras complejas de objetos.

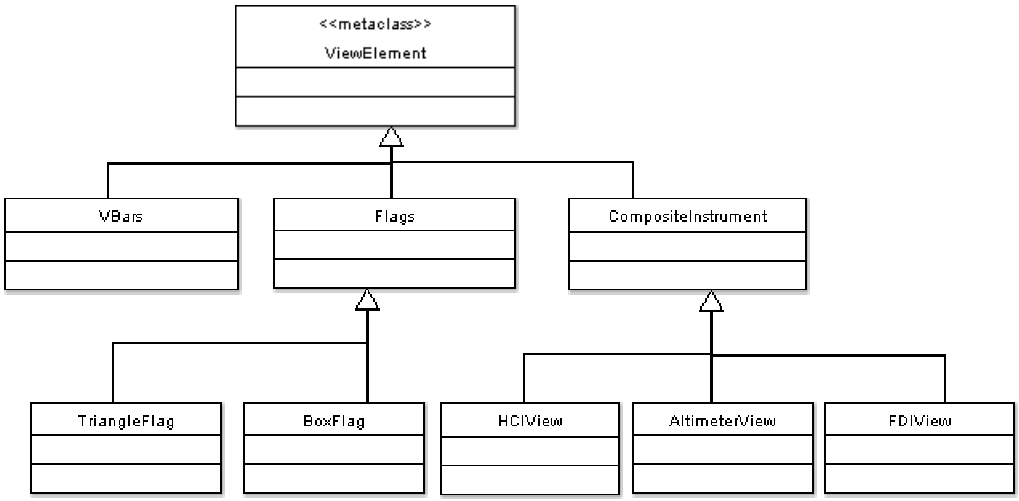


**Figura 48 Cabina de Vuelo – Est. Interna**

Para la creación de un tipo de cabina dado o inclusive para representar marcas determinadas de instrumentos se utiliza el patrón ***Builder*** [GAM95] el cual nos permitirá cambiar el diseño de la cabina acorde a la aeronave que se desee simular ya que el algoritmo de construcción puede generalizarse y crear diferentes representaciones basados en este algoritmo.

Por lo tanto tendremos:

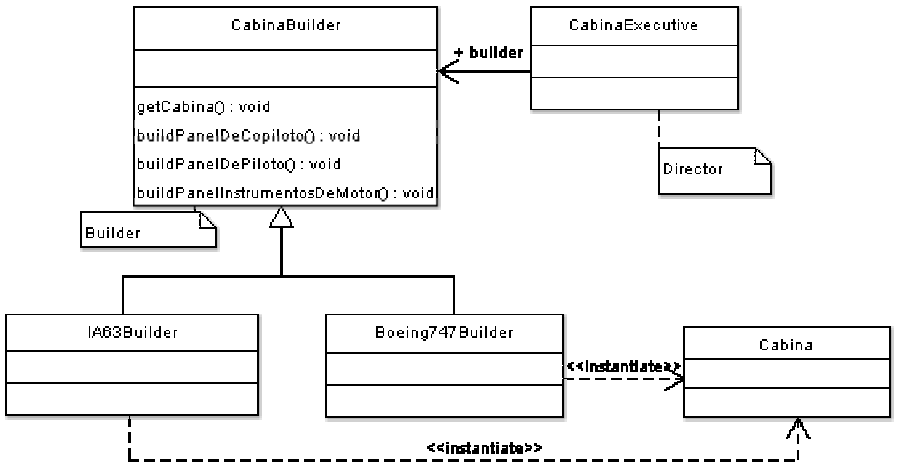


**Figura 49 Builder de Cabina de Vuelo**

## 8.3 - Instumentos Reales

Estan relacionados a la abstracción del hardware. La interface es la del instrumento pero se encapsula la implementación del uso. Esto es, dentro del MVC presentado anteriormente, solo se presenta el modelo de instrumento y utilizamos el patrón ***Bridge*** [GAM95] para abstraer la implementación del mismo.

Entonces de lo planteado hasta el momento tenemos que un instrumento totalmente simulado tendrá la siguiente estructura:

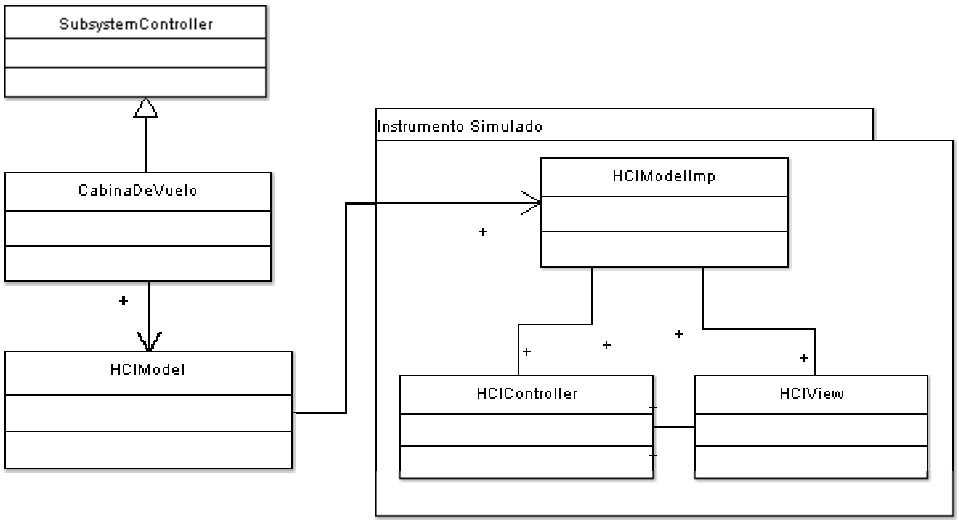


**Figura 50 Instrumento Simulado - Estructura**

Para el caso de instrumentos reales tendremos:

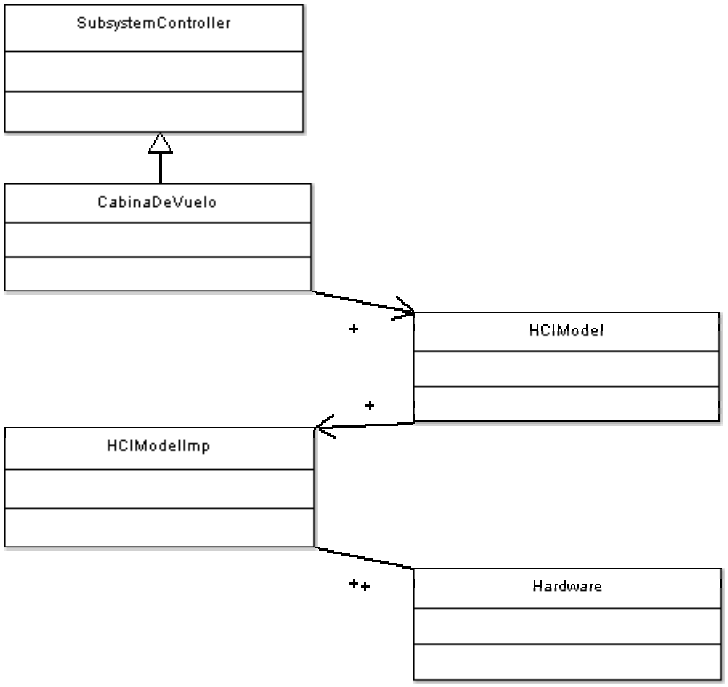


**Figura 51 Instrumento Real - Estructura**

Esto será visto en mayor profundidad cuando se trate el tema de Abstracción del Hardware.

## 8.4 - Comandos

Las acciones que realiza el usuario, podemos dividirlas en dos grandes grupos: en el primero agrupamos a todas aquellas acciones que afectan al modelo del instrumento como puede ser fijar el nivel del suelo en el altímetro. Esto implica fijar un parámetro que sólo afecta y es visto por el modelo del instrumento y que no afecta a la simulación en si misma. Entonces este tipo de acciones son las comúnmente manejadas por el Controller del MVC.

La segunda clase de acciones son las que afectan la simulación como puede ser el cambio de posición del bastón de mando de la aeronave. Este tipo de acciones ingresará directamente a la simulación (hardware) y se reflejará en un cambio de estado que será transmitido e interpretado en la cabina de vuelo.

## 8.5 - Conclusión

Nótese que con la solución adoptada, la cabina de vuelo queda modelada con un ***Facade*** [GAM95] a través de un ***Distributed Proxy*** [SIL97], que con el patrón ***Builder*** [GAM95] agrega instrumentos reales y/o simulados por software, siguiendo el patrón estructural para el cual representa un subsistema.

En la figura se ve una muestra de instrumentos simulados utilizando los modelos presentados:


**Figura 52 Instrumentos Simulados (GSDV)**

Nótese entonces que por ejemplo los instrumentos indicados como THR, RPM, etc. (derecha e izquierda del HCI) son cada uno, una instancia de la misma clase. A su vez esta clase está compuesta (patrón composite) por otras instancias de clases como el cuadrante de la escala.

Los instrumentos bajo este modelo tiene la gran ventaja que se pueden realizar estudios como por ejemplo: Estudiar las respuestas del piloto si en vez de representarse el ángulo de ataque en el FDI, se le presenta el ángulo de incidencia. Este cambio para el estudio se realizaría de una forma muy sencilla ya que implica solamente cambiar la variable angular solicitada como entrada del FDI.

# Capítulo 9 Modelo Ambiental

Hasta el momento hemos planteado los modelos que componen a la aeronave como sistema, sus subsistemas y la interfaz con el humano. Ahora, es de poca utilidad y no posee demasiado sentido la simulación de una aeronave sin incorporar los factores externos que la afectan. Es por esto que en el presente capítulo vamos a plantear la modelización del ambiente en el que está inmersa la aeronave, o sea, "todo lo externo".

## 9.1 - Planteo General

El ambiente se encarga de simular el mundo que rodea a la aeronave. Así corresponde a este modelo la simulación de:

- Modelo gravitacional
- Atmósfera
- Vientos
- Obstáculos (montañas, edificios, puentes, etc.)
- Navegación (radio faros, aeropuertos, etc.)
- Y todo aquello que constituya parte del ambiente en que se encuentra inmersa la aeronave.

Si tomamos el modelo gravitacional, acorde al propósito de la simulación, se deseará implementar un modelo de gravedad constante; el modelo de la tierra tomada como esfera; modelo de la tierra ovalada; etc. Para la atmósfera existen varios modelos como por ejemplo el de densidad constante.

Entonces para cada uno de estos casos estamos en presencia de variación en los algoritmos de cálculo y por lo tanto el patrón a aplicar es el ***Strategy [GAM95]*** que permite definir una familia de algoritmos

intercambiables, permitiendo la variación independiente del cliente que los utiliza.

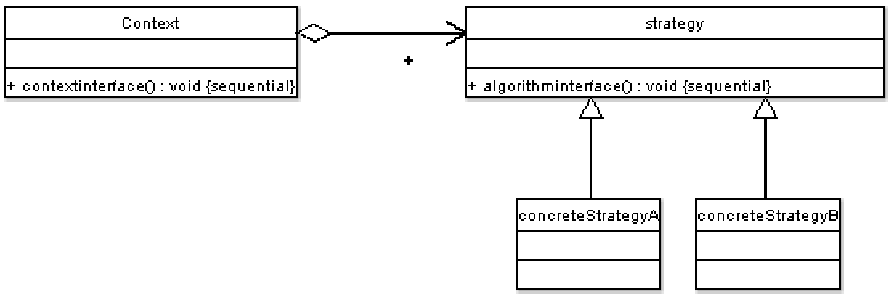


**Figura 53 Strategy Pattern**

Entonces tendremos:

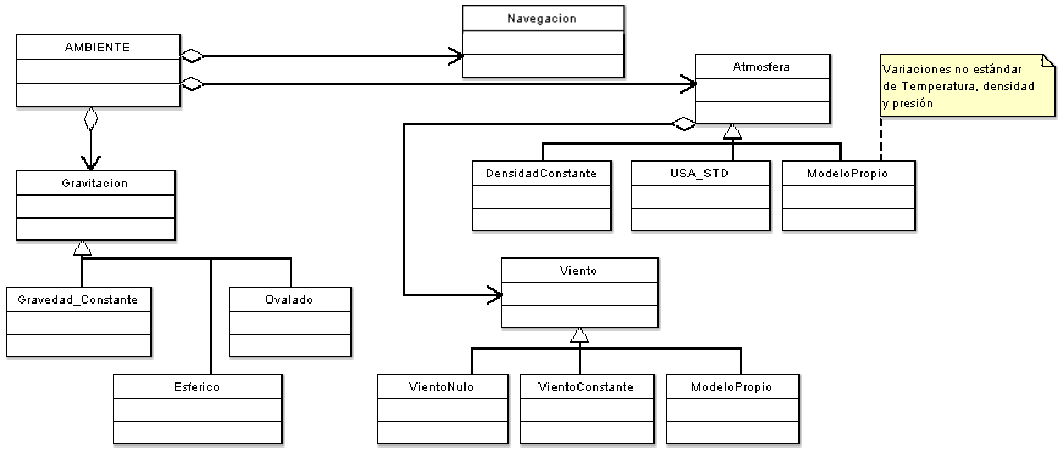


**Figura 54 Modelo Ambiental**

Por lo tanto con el patrón ***Strategy [GAM95]*** se obtiene una mejora en el modelo de atmósfera, gravitacional y navegación.

Para el caso del modelo de atmósfera el patrón ***Composite [GAM95]*** nos permite generar modelos de vientos complejos, por ejemplo combinando modelos de vientos constantes con vientos en ráfaga. Para esto la reforma en el gráfico anterior sería:

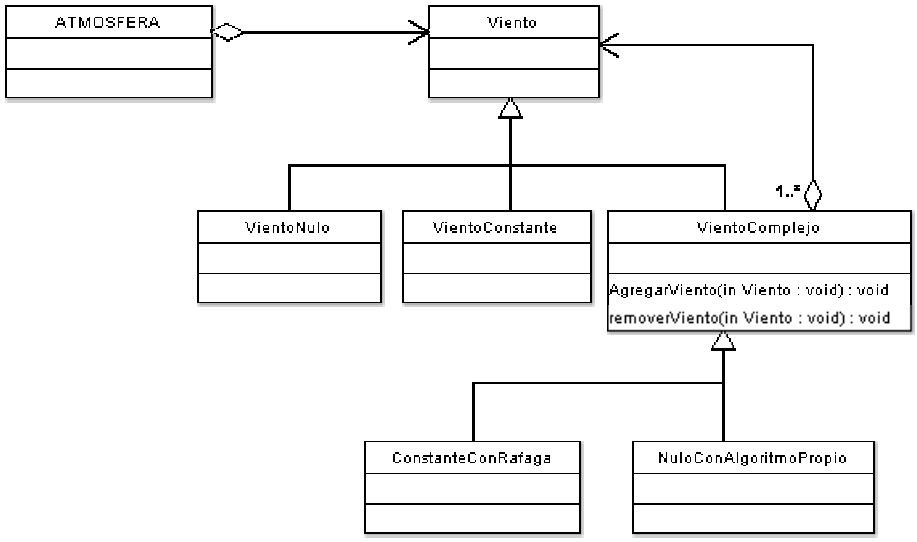


**Figura 55 Modelo Atmosférico**

Otro patrón a utilizar aquí es el ***Random Event Generation*** [SIM94] que genera eventos para realmente lograr una simulación de la naturaleza. La distribución de probabilidad se puede obtener por una ley binomial de Poisson, beta, gamma, Erlang, Exponencial, Weibull, Normal, logarítmica, uniforme y triangular. La función determina el valor del tiempo para la próxima ocurrencia del evento.

Si Tomamos ahora el subsistema denominado Navegación, tendremos que la posición de la aeronave en el mundo estará controlada por el Sistema de Posicionamiento Global (GPS), el cual realizará los cálculos referentes a la posición respecto a la terna inercial seleccionada. Esto nos permitirá ubicarnos dentro del mundo simulado para poder generar inclusive relaciones con los objetos que componen el ambiente, como aeropuertos, estaciones de VOR, etc.

Con esto plantearemos un modelo básico para el sistema de navegación según se muestra en la figura.

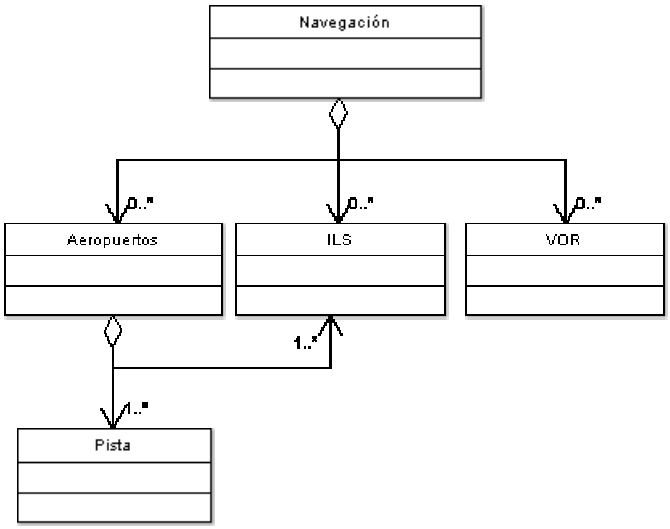


**Figura 56 Modelo Navegación**

Los distintos parámetros que se generen en los cálculos de la dinámica de la aeronave, alimentarán al modelo ambiental, y a su vez, este brindará datos que serán utilizados tanto en los cálculos como por los instrumentos de cabina, como es el caso del VOR e ILS.

# Capítulo 10 Conclusiones y Trabajo Futuro

En la presente tesis hemos planteado la aplicación de patrones de diseño y modelos orientados a objetos para lograr la simulación del vuelo de aeronaves de forma que sea adaptable, fácil de mantener y permita al ingeniero Aeronáutico concentrarse en los problemas del diseño de aeronave bajo estudio. Se han presentado las estructuras utilizadas actualmente y se ha planteado el modelo patrón de diseño del tipo estructural (AVS) que generamos a partir de este estudio. Además hemos desarrollado cada parte del simulador con un planteo basado en objetos y aplicando patrones de diseño existentes para mejorar este planteo, ya que son soluciones probadas que ayudan a mejorar la calidad del software. De esta forma hemos cubierto la aeronave como sistema pero también los sistemas externos como el medio ambiente que, generalmente, no se encuentran desarrollados en la bibliografía.

Este trabajo entonces constituye un aporte al dominio de la simulación en el campo aeronáutico ya que se ha establecido los objetos que definen el dominio; se han definido modelos y patrones de diseño para lograr flexibilidad, calidad y reuso; se han desarrollado aspectos del dominio que generalmente no son tratados dentro del mismo contexto que al modelo de la aeronave.

El uso de OOA en los modelos desarrollados, nos permite una codificación modular, esconder los detalles de implementación y crear una estructura de código jerárquica logrando los objetivos del presente trabajo: “lograr modelos que constituyan al software del simulador fácil de mantener, pasible de reuso y fácil de evolucionar”.

## 10.1 - Trabajo Futuro

La amplitud del tema y el espectro de una tesis no nos permite analizar todos los aspectos abarcados en la simulación de aeronaves y por temas pendientes podemos mencionar:

- ✓ **Seguridad y confiabilidad:** Análisis de patrones para seguridad y confiabilidad del software [DOU02] ya que es imprescindible que si se está analizando procesos de vuelo de varias horas o bien se están entrenando tripulaciones, no pueden haber problemas en el funcionamiento del simulador.
- ✓ **Scheduling:** Otro aspecto pendiente se relaciona con el scheduling y el análisis de patrones a utilizar como podrían ser Asynchronous Task Completion Pattern; combinación de Synchronizer con Observer, etc.
- ✓ **Certificación:** La tecnología orientada a objetos actualmente se encuentra poco presente en aviación para sitemas de computadoras de abordo, a pesar que la tecnología orientada a objetos se asume promueve la productividad, reusabilidad y mejora la calidad del software, incluso existen trabajos que están contra su aplicación como en [HAY03]. El mayor obstáculo ha sido el hecho de la certificación, por lo cual se ha publicado el "Handbook for OO Technology in Aviation" (OOTIA), 26/10/2004, como una forma de plantear y comenzar a debatir esta problemática. Este tipo de modelos OO pueden ayudar a resolver este tipo de problemas ya que el sistema OO a incluir en la aeronave real, podría ser perfectamente verificado y certificado con la ayuda de este tipo de tecnología.

# Capítulo 11 Referencias Bibliográficas

### 11.1.1 -Bibliografía utilizada en la presente Tesis

En este punto se presenta la bibliografía específica utilizada para el desarrollo de la presente tesis.


1. [APP01] Patterns in a Nutshell: The “bare essentials” of Software Patterns; Brad Appleton; http://www.enteract.com/~bradapp/.
2. [TA01] Patrones de Diseño: Reutilización de ideas; Alberto López Tallón, 1998 http://www.aqs.es/web/files/designpatterns.html#1.
3. [COO98] The Design Patterns – Java Companion; James W. Cooper; Addison-Wesley, 1998.
4. [SK01] Simulation that is used in several Operations Research related courses taught at the Maltese University; J. Sklenar; University of Malta, 2000.
5. [CH01] Simulación en el Ejército: El desafío que viene, Chile, 1999.-
6. [EKS01] Design Patterns for Simulations in Erlang/OTP; Ulf Ekstrom; Information Technology Computing Science Department; Uppsala University, Uppsala, Sweden
7. [BASS03] Software Architecture in Practice, Second Edition; Len Bass, Paul Clements, Rick Kazman; Addison Wesley, 2003; ISBN : 0-321-15495-9.
8. [CRI93] DARTS: A DOMAIN ARCHITECTURE FOR REUSE IN TRAINING SYSTEMS; Robert G. Crispen, Brett W. Freemon, K. C. King, and William V. Tucker; Boeing Defense & Space Group, Huntsville, Alabama - 1993.
9. [KAZ01] Distributed Flight Simulation: A Challenge for Software Architecture; Rick Kazman, Department of Computer Science, University of Waterloo, Waterloo, Ontario
10. [BASS93] Structural Modeling: An Application Framework and Development Process for Flight Simulators; Gregory D. Abowd, Len Bass, Larry Howard, Linda Northrop, August 1993. Technical Report CMU/SEI-93-TR-14
11. [CHA96] A Case Study in Structural Modeling, Gary Chastek, Lisa Brownsword, December 1996
12. [BRO98] The Design Patterns Smalltalk Companion, Alpert, Brown, Woolf. ADDISON WESLEY, ISBN 0-201-18462-1, 1998
13. [GAM95] Design Patterns, Elements of Reusable Object Oriented Software. Gamma, Helm, Johnson, Vlissides. ADDISON WESLEY. ISBN 0-201-63361-2. 1995.
14. [LAM96] Build Your Own Flight Sim in C++; Radtke, Lampton; THE WAITE GROUP; ISBN 1-57169-022-0; 1996.

15. [SIL96] Object Synchronizer: A Design Pattern for Object Synchronization; Silva, Pereira, Marques; EuroPLoP96, Kloster Irsee, Germany, July, 1996.

16. [BUHR93] Designing with Timethreads. R. J.A. Buhr, R.S. Casselman. Department of Systems & Computer Engineering. Carleton University. 1993

17. [SIL97] Distributed Proxy: A Design Pattern for Distributed Object Communication. Ant´onio Rito Silva, Francisco Assis Rosa, Teresa Gonc¸alves, Miguel Antunes. Allerton Park, Illinois, USA, September 1997.

18. [COA92] Object-oriented patterns. COPYRIGHT Association for Computing Machinery. Inc. 1992. Peter Coad

19. [CASS93] A Role-Based Architectural Model Applied to Object-Oriented Systems. Ronald S. Casselman, B.Eng.. Master of Engineering Thesis. Ottawa-Carleton Institute for Electrical Engineering, 1993.

20. [SCH98] Acceptor-Connector. Douglas C. Schmidt 1998, all rights reserved, © Siemens AG 1998, all rights reserved.

21. [PASQ01] Technologies for Distributed Simulation: CORBA and HLA. Antonello Pasquarelli1. Information Technologies of Alenia Marconi Systems, Rome.

22. [EVH03] Hardware Device Design Pattern. EventHelix.com Inc. 2003

23. [EVH02] Manager Design Pattern. EventHelix.com Inc. 2003

24. [CLA00] Advanced Distributed Simulation for the Australian Defence Force. Peter Clark, Peter Ryan and Lucien Zalcman. Air Operations Division. Aeronautical and Maritime Research Laboratory. DSTO-GD-0255. 2000.

25. [UHR96] Concepts of Object- and Agent-Oriented Simulation. Adelinde M. Uhrmacher, Artificial Intelligence Laboratory, University of Ulm, 1996

26. [CHA01] Advancing the State of the Art in Flight Simulation via the Use of Synthetic Environments. Frank R. Chavez, Jim Bernard, , Iowa State University

27. [CARL95] SIM94 – A concurrent Simulator for Plan-Driven Troops. Carlsons y Tronje. Computing Science Department, Uppsala University. Sweden. 1995.

28. [CAD90] Next-Generation Architecture to Support Simulation-Based Acquisition. Cadhda, Welsh. Loockheed Martin ATL. 1990.

29. [LEA95] Design Patterns for Avionics Control Systems. Doug Lea, SUNY Oswego & NY CASE Center, DSSA Adage Project ADAGE-OSW-94-01, DRAFT Version 0.9.6; November 20, 1994 (reformatted March, 1995).

30. [ENR02] A Flight Software Development and Simulation Framework for Advanced Space Systems. John P. Enright, David W. Miller. May 2002. Doctor of Philosophy Thesis at the Massachusetts Institute of Technology.

31. [HAR95] Robust Flight Control: A Real-Time Simulation Investigation. Harman, Liu. Institute for Aerospace Studies. University of Toronto, Ontario, Canada. 1995.

32. [ROU97] Application of a Multipurpose Simulation Design. Bell, O'Rourke. Bihrle Applied Research, Inc. 1997. AIAA-97-3798.

33. [LES01] SIMULATION OF A F/A-18 E/F DROP MODEL USING THE LaSRS++ FRAMEWORK. Kevin Cunningham, P. Sean Kenney, Richard A. Leslie, David W. Geyer, Michael M. Madden, Patricia C. Glaab. Unisys Corporation. NASA Langley Research Center. AIAA-98-4160

34. [LES02] LaSRS++ AN OBJECT-ORIENTED FRAMEWORK FOR REAL-TIME SIMULATION OF AIRCRAFT. Richard A. Leslie, David W. Geyer, Kevin Cunningham, Patricia C. Glaab, P. Sean Kenney, Michael M. Madden, Unisys Corporation, NASA Langley Research Center. AIAA-98-4529

35. [MAD01] CONSTRUCTING A MULTIPLE-VEHICLE, MULTIPLE-CPU SIMULATION USING OBJECT-ORIENTED C++. Michael M. Madden*, Patricia C. Glaab, Kevin Cunningham*, P. Sean Kenney, Richard A. Leslie, David W. Geyer, Unisys Corporation. AIAA-98-4530

36. [MAD02] USING ABSTRATION TO ISOLATE HARDWARE IN AN OBJECT-ORIENTED SIMULATION. P. Sean Kenney, Richard A. Leslie, David W. Geyer, Michael M. Madden*, Patricia C. Glaab, Kevin Cunningham*. Unisys Corporation, NASA Langley Research Center, AIAA-98-4533.

37. [MAD03] MANAGING SHARED MEMORY SPACES IN AN OBJECT-ORIENTED REAL-TIME SIMULATION. David W. Geyer, Michael M. Madden, Patricia C. Glaab, Kevin Cunningham, P. Sean Kenney, Richard A. Leslie. AIAA-98-4532

38. [CUN99] USE OF THE MEDIATOR DESIGN PATTERN IN THE LaSRS++ FRAMEWORK. Kevin Cunningham. AIAA-99-4336

39. [GEY99] The Use of Multiple Threads in An Object-Oriented Real-Time Simulation. David W. Geyer. AIAA-99-4338

40. [GLA99] A GENERIC OBJECT-ORIENTED IMPLEMENTATION FOR FLIGHT CONTROL SYSTEMS . Patricia C. Glaab, Michael M. Madden* AIAA-99-4339

41. [KENN99] USING ABSTRACTION TO CREATE A PORTABLE OBJECT-ORIENTED SIMULATION. P. Sean Kenney and David W. Geyer. AIAA-99-4340

42. [AIA00] A GENERIC LANDING GEAR DYNAMICS MODEL FOR LASRS++, AIAA-2000-4303

43. [AIA4388] AN OBJECT-ORIENTED DESIGN FOR TRIM. AIAA-2000-4388

44. [AIA4390] MANAGING THE PLAYBACK AND RECORDING OF SIMULATION MODELS IN AN OBJECT-ORIENTED SIMULATION. AIAA-2000-4390

45. [AIA4391] AN OBJECT-ORIENTED INTERFACE FOR SIMULINK MODELS. AIAA-2000-4391

46. [AIA4393] MPLEMENTING DYNAMIC SYSTEM MODELS IN THE ASSET SIMULATION FRAMEWORK. AIAA-2000-4393

47. [AIA4500] DATA MANAGEMENT IN THE ASSET SIMULATION FRAMEWORK. AIAA 2000-4500.

48. [AIA4057] A UNIQUE SOFTWARE SYSTEM FOR SIMULATION-TO-FLIGHT RESEARCH. AIAA 2001-4057

49. [AIA4119] EXAMINING REUSE IN LASRS++-BASED PROJECTS. AIAA-2001-4119

50. [AIA4123] AN OBJECT-ORIENTED SENSOR AND SENSOR SYSTEM DESIGN. AIAA 2001-4123.

51. [AIA4244] PLATFORM-INDEPENDENCE AND SCHEDULING IN A MULTI-THREADED REAL-TIME SIMULATION. AIAA-2001-4244.

52. [NLRC99] RESEARCH FLIGHT MANAGEMENT SYSTEM B-757 BASELINE HARDWARE CONFIGURATION. NASA LANGLEY RESEARCH CENTER. DECEMBER 15, 2000

53. [KUD99] Design And Development of Lateral Flight Director. Kudlinkski, Ragsdale. NASA. 1999

54. [KAP99] Automated Concurrent Blackboard System Generation in C++. Kaplan, McManus, Bynum. NASA. 1999.

55. [BLA01] A Case Study: Designing a Discrete Event Simulation. David Blackledge.

56. [RYA00] A Pattern Language for Efficient, Predictable, Scalable, and Flexible Dispatching Mechanisms for Distributed Object Computing Middleware. Irfan Pyaraliand Carlos O'Ryan, Department of Computer Science, Washington University, Douglas C. Schmidt, Electrical and Computer Engineering Dept. 2000.

57. [EIN01] Using Inner Classes in Design Patterns. Daniel Einarson and Görel Hedin. Dept of Computer Science, Lund University.

58. [WAS92] Modeling and Model Simplification of Aeroelastic Vehicles: An Overview; Martin R. Waszak and Carey S. Buttrill; Langley Research Center; David K. Schmidt; Arizona State University; 1992.

59. [REH95] A Handbook of Flight Simulation Fidelity Requirements for Human Factors Research; Albert J. Rehmann; NASA; 1995.

60. [CRISPEN01] Structural Model: Architecture for Software Designers. Robert G. Crispen and Lynn D. Stuckey, Jr.; Boeing Defense & Space Group; Huntsville, Alabama 35824.

61. [YOU94] Object-Oriented Systems Design, An Integrated Approach. Edward Yourdon; Yourdon Press Computing Systems, 1994.

62. [SJO02] Design of a flight Simulator Software Architecture. Karl-Erik Sjöquist. School of Mathematics and System Engineering. MSI, Växjö University, 2002. SE-351 95 VÄXJÖ. Suecia.

63. [SEL01] An Architectural Pattern for Real-Time Control Software. Bran Selic. ObjectTime Limited. Kanata, Ontario, Canada.

64. [KEG01] Application Of Patterns To Real-Time Object Oriented Software Design. Master Thesis. Queen's University, Kingston, Ontario, Canada.

65. [DOU02] Real-Time Design Patters: Robust Scalable Architecture for Real-Time Systems. Addison Wesley, 2002. ISBN: 0-201-69956-7.

66. [HAY03] Considering Object Oriented Technology in Aviation Applications. Kelly J Hayhurst, C Michael Holloway. NASA Langley Research Center. NASA 2003-22.

### 11.1.2 -Bibliografía Aeronáutica

La presente bibliografía constituye una referencia teórica sobre el dominio aeronáutico, el cual obviamente es el fenómeno físico a simular. La bibliografía cubre, especialmente, los conceptos de estabilidad y control de aeronaves, los cuales, en la construcción de simuladores, se necesitarán dominar.


[A1]. Aerodinámica en Preguntas y Respuestas - N.F. Crasnov - MIR. 1995.

[A2]. Aerodynamics Components at High Speeds. Vol I - Edward, Garrick & Frick – NASA - 1956.

[A3]. Aerodynamics Components at High Speeds. Vol II - Edward, Garric, Fick, Ferrari, Smith, Jones, Kohen, Brown - Pricenton University Press - 1957

[A4]. Aerodynamics for Engineering Students - E.L. Houghthon & N.B. Carruthers - Thomson Press - 1984.

[A5]. Aerodynamics for Engineering Students - E.L. Houghton & P.W. Carpenter. J. Wiley - 1993.

[A6]. Aerodynamics for Engineers - John J. Bertin - Prentices Hall - 1990.

[A7]. Aerodynamics of the Airplane - Hermann Schlichting – Mc. Graw Hill – 1979.

[A8]. Aerodynamics Theory: Airplane as a Whole, Aerodynamics of the Airships, etc. Vol VI – W.F. Durand – Peter Smith – 1986.

[A9]. Arodynamics Theory: Applied Airfoil Theory. Vol IV – W.F. Dura – Peter Smith – 1986.

[A10]. Arodynamics Theory: Dynamics of the Airplane, Airplane Performance. Vol V – W.F. Dura – Peter Smith – 1986.

[A11]. Arodynamics Theory: General Aerodynamic. Theory Perfect Fuids. Vol II – W.F. Dura – Peter Smith – 1986.

[A12]. Arodynamics Theory: mathematical Aids. Fuild Dynamics. Historical Sketch. Vol I – W.F. Dura – Peter Smith – 1986.

[A13]. Aerodynamics, Aeronautics and Flight Mechanics – B. Mac Cormick Jr. – J. Wiley – 1995.

[A14]. Aeroelasticity – Bisplinhoff, Ashley, Halman – Dover – 1996.

[A15]. Aircraft Control and Simulation – B.L. Stevens, F.L. Lewis – J.Wiley – 1992.

[A16]. Aifcraft Dynamic and Automatic Control – Mac Ruer Askenas, Graham – Princeton – 1979.

[A17]. Aricraft Dynamic Stability and Response – A.W. Babister – Pergamon Press – 1980.

[A18]. Aircraft Handling Qualities – J.Hodgkinson – A.I.A.A. – 1998.

[A19]. Aircraft Performance – W.A. Mair & D.L. Birdsal – Cambridge University Press – 1996.

[A20]. Aircraft Stability and Control – A.W. Babister – Pergamon Press – 1961.

[A21]. Airplane Aerodynamics and Performance – Chuan Tau, Edward Lan, J. Roskam – Roskam Aviation and Engineering Corporation – 1991.

[A22]. Airplane Design (Part VII) Determination of Stability, Control and Performances. Characteristics for FAR and MIL requirements – J. Roskam – Roskam Aviation and Engineering Corporation – 1991.

[A23]. Airplane Design Volumen VII – J. Roskam – Roskam Aviation and Engineering Corporation .- 1992.

[A24]. Airplane Filight Dynamics and Automatic Flight Control – J. Roskam – Roskam Aviation and Engineering Corporation – 1994.

[A25]. Airplane Performance Selection and Design – Francis Hale – J. Wiley – 1984.

[A26]. Airplane Performance Stability and Control – C.D. Perkins, R.E. Hage – J.Wiley & Son – Ed. 1949, Reeditado 2000.

[A27]. Automatic Control Engineering – F.H. Raven – Mc Graw Hill – 1995.

[A28]. Automatic Control of Aircraft and Misiles – J. Blakelock – J. Wiley – 1991.

[A29]. Automatic Flight Control System – D. Mc. Lean – Prentice Hall – 1990.

[A30]. Basic Wing and Airfoil Theory – A. Pope – Mc. Graw Hill – 1951.

[A31]. Components Weights Estimation. Airplane Design. Vol V – J. Roskam – Roskam Aviation and Engineering Corporation – 1991.

[A32]. Computer Assisted Analysis of Aircraft Performance Stability and Control – F. Smetana – Mc Graw Hill – 1984.

[A33]. Control de Sistemas Dinámicos con Retroalimentación – G. Franklin, J. Powell, A. Emami Naemi – Addison Wessley – 1991.

[A34]. Determinación del Vector Tracción sobre los discos de Hélices y Fans – V. Caballini y otros – Grupo de Simulación Dinámica del Vuelo. UTN FRH – 1992.

[A35]. Determination of Stability & Control, and Performance – J. Roskam – Roskam Aviation and Engineering Corporation – 1992.

[A36]. Distribución de Sustentación en Alas con Asimetrías Aerodinámica, Geométrica y Funcional – Grupo de Simulación Dinámica del Vuelo. UTN FRH – 1993.

[A37]. Dynamic of Flight Stability and Control – B. Etkin – J. Wiley – 1982.

[A38]. Dynamic of Atmospheric Flight – B. Etkin – J. Wiley – 1972.

[A39]. Dynamic of Helicopter Flight – G.H. Saunders – J. Wiley – 1975.

[A40]. Engineering Supersonics Aerodynamics – E.A. Bonney – MC Graw Hill – 1950.

[A41]. Elements of Aerodynamics of Supersonic Flows – A. Ferri – Mac Millan Company

[A42]. Flight Dynamic of Rigid Airplanes, an introduction to – G. Hancoc – Elis Horwood Series – 1995.

[A43]. Flight Dynamics of Rigid and Elastic Airplanes – J. Roskam – Roskam Aviation and Engineering Corporation – 1975.

[A44]. Flight Mechanic of High Performance Aircraft. Book 4 – N.X. Vinh – Cambridge University Press Series – 1993

[A45]. Flight Peformance of Aircraft – Shiva Kumar Ojha – A.I.A.A. – 1998.

[A46]. Flight Simulation – J.M. Rolfe, K.J. Staples – Cambrich Aeroespace Series – 1994.

[A47]. Flight Stability and Automatic Control – R.C. Nelson – Mc Graw Hill – 1989.

[A48]. Flying Qualities and Flight Testing of Airplanes – D. Stinton – A.I.A.A. – 1997.

[A49]. Fundamentals of Aerodynamics – John D. Anderson, Jr. – Mc Graw Hill – 1991.

[A50]. Fundamentos de Servotécnica – J.G. Barquero – Phillips – 1971.

[A51]. Fundations of Aerodynamics, Bases of Aerodynamic Design – A.M. Kuethe, Chuen Yen Chow – J. Wiley & Son – Ediciones Varias.

[A52]. Helicopter Performance Stability and Control – R. Prouty – Krieger Publishing Co – 1995.

[A53]. Helicopter Theory – Wayne Johnson – Dover -1980.

[A54]. High Speeds Aerodynamics – E. Carafoli – Pergamo Press.

[A55]. Introduction to Aeronautical Dynamics – M. Rauscher – J. Wiley – 1953.

[A56]. Introduction to Aeronautics: A Design Perspective – S.A. Rand, R.J. Stiles, J.J. Bertin, R. Whitford – A.I.A.A. – 1997.

[A57]. Introduction to Aircraft Flight Dynamics – L.V. Schmidt – A.I.A.A. – 1998.

[A58]. Introduction to Space Dynamics – W.T. Thompson – Dover – 1986.

[A59]. Low Speeds Aerodynamics. From Wings Theory to Panel Methods – J. Katz, A. Plotkin – Mc Graw Hill – 1991.

[A60]. Modern Spacecrafts, Dynamic and Control – M.H. Kaplan – J. Wiley – 1976.

[A61]. Preliminay Calculation of Aerodynamic Thrust and Power Characteristics. Airplane Design. Tomo VI – J. Roskam – Roskam Aviation and Engineering – 1991.

[A62]. Preliminay Configuration Design and Integration of the Propulsion System. Airplane Design – J. Roskam – Roskam Aviation and Engineering Corporation.

[A63]. Preliminary Sizing of the Airplane. Airplane Design. Tomo I – J. Roskam – Roskam Aviation and Engineering Corporation – 1991.

[A64]. Performance, Stability, Dynamics and Control of Airplanes – B.N. Pamadi – A.I.A.A. – 1998.

[A65]. Principles of Aeroelasticity – R. Blisplinghoff, H. Ashley – Dover – 1975.

[A66]. Rotary Wing. Aerodynamic – W.Z. Stepniewski, C.N. Keys – Dover – 1980.

[A67]. Stability and Control of Airplanes and Helicopters – E. Seckel – Academic Press – 1964.

[A68]. Teoría del Ala y la Hélice – H. Glauert – I.N.T.A. – 1946.

[A69]. The Aerodynamic Design of Aircraft – D. Kuchemann, FRS – Pergamon Press – 1985.

[A70]. Theorical Aerodynamics – L.M. Milne Thomson – Dover – 1983.

[A71]. Theory of Aerostability. An introduction to – Y.C. Fung – Dover – 1993.

[A72]. Theory of Flight – R. Von Mises – Dover – Ediciones Varias.

[A73]. Theory of Wing Sections – I.H. Abbott, A.E. Von Doenhoff – Dover – Ediciones Varias.