Desarrollo de un Sistema de Debriefing para la Fuerza Aérea Argentina siguiendo la Metodología de Proceso Unificado

Trabajo de posgrado para optar al título de
Especialista en Gestión de la Innovación y la Vinculación Tecnológica

Estudiante:

Juan Pablo Rumie Vittar

Director:

Gustavo Rodriguez, MSc.

Universidad Nacional de Rosario
Centro de Estudios Interdisciplinarios
Especialización en Gestión de la Innovación y la Vinculación Tecnológica

2024

A Dios por estar en todo lo que tengo a mi alrededor.

A toda mi Familia (Padres, Hermanos, Sobrinos, Tíos, etc) por ser parte fundamental de mi vida y que de alguna u otra manera han contribuido para el logro de mis objetivos.

Al Centro de Investigación y Desarrollo de Tecnologías Aeronáuticas y a la Dirección General de Investigación y Desarrollo de la Fuerza Aérea Argentina y a todo su personal por acompañarme en esta formación personal.

Al Centro de Estudios Interdisciplinarios de la Universidad Nacional de Rosario, y todo el grupo de docentes y no docentes que la conforman,

y a mi Argentina, que a través de su sistema de educación pública tengo la oportunidad de crecer día a día profesionalmente y contribuir así al avance tecnológico/científico del país...

MUCHAS GRACIAS!

Índice general

Índice de figuras

Índice de tablas

Acrónimos

ADS-B Automatic Dependent Surveillance – Broadcast.

BIT Build In Test.

CEV Centro de Ensayo en Vuelo.

CITeA Centro de Investigación y Desarrollo de Tecnologías Aeronáuticas.

DEM Digital Elevation Model.

DEyH Dirección de Evaluación y Homologación.

DGID Dirección General de Investigación y Desarrollo.

EAM Escuela de Aviación Militar.

FAA Fuerza Aérea Argentina.

GPS Global Positioning System.

GV Generador de Vuelos del Sistema de Debriefing.

I+D Investigación y Desarrollo.

NGA National Geospatial-Intelligence Agency.

nube Internet.

PC Personal Computer.

PNG Portable Network Graphics.

RC Release Candidate.

RO Requerimiento Operativo.

SARM Sistema de Armas.

SD Sistema de Debriefing.

SOH State of Health.

SRPV Sistema Registrador de Parámetros de Vuelo.

SRPVP Sistema Registrador de Parámetros de Vuelo Portátil.

SW Software.

VyV Validación y Verificación.

Capítulo 1

Introducción

1.1. Resumen

En Argentina, numerosas empresas incorporan componentes electrónicos en sus productos y operaciones. Estas industrias muestran un creciente interés en adoptar tecnología para mejorar la eficiencia de producción, disminuir la dependencia de importaciones y aumentar el valor agregado de sus productos para la exportación. En este contexto, muchas empresas se ven obligadas a implementar o desarrollar sistemas con certificación de seguridad funcional (IEC 61508, 2010). Esto se realiza con el fin de salvaguardar a las personas, el medio ambiente, las inversiones y, en muchos casos, para cumplir con los exigentes requisitos de exportación. En sectores como la aeronáutica y la industria espacial, donde se crean sistemas críticos con hardware y software complejos, este proceso es especialmente crucial (RTCA DO-178C, 2011; RTCA DO-278, 2011; SAE ARP 4754B, 2023; SAE ARP 4761A, 2023). Estos sistemas suelen tener ciclos de vida prolongados que pueden extenderse a lo largo de décadas (IEEE/EIA 12207, 2017). El diseño de estos sistemas, tanto en hardware como en software, constituye la fase principal del proyecto, siendo las pruebas y la documentación derivadas de estos modelos iniciales un componente esencial del proceso. El objetivo de este trabajo es desarrollar un Sistema de Monitoreo de Parámetros de Vuelo (utilizando la metodología de desarrollo de SW denominado Proceso Unificado (Ivar Jacobson y Rumbaugh, 1999)) que permita la visualización en simultaneo de diferentes entradas, tanto propias de aeronaves como así también

de sistemas registradores en vuelo, tales como, el Dispositivo Electrónico de Parámetros de Vuelo (CITeA SRPV).
Este sistema de monitoreo (de aquí en más denominado SW) visual sera diseñado, desarrollado, verificado y mantenido por personal del CITeA y el proceso de homologación será llevado a cabo por la Dirección de Evaluación y Homologación dependiente de la Dirección General de Investigación y Desarrollo para de esta manera, cumplir con los estándares de seguridad funcional que exigen las normativas vigentes en las Fuerzas Armadas. El SD es un Software Operativo e Interactivo de visualización y análisis de ejercicios pos vuelo que ofrece la posibilidad de llevar a cabo la reproducción y de los mismos registrados mediante un Generador de Vuelos (GV). Cuenta con la capacidad de reproducir hasta 20 aeronaves de forma simultánea provenientes de diversas fuentes de datos. Estas incluyen sistemas ADS-B, SRPV, SARM IA-63 Pampa II, SARM A4AR, GPS en SARM IA-58 Pucará.
Las funciones principales del sistema son las siguientes:

- Representación de los datos de vuelo en 2D y 3D.
- Capacidad de sincronización rápida basada en una selección de salto a un tiempo específico por parte del usuario.
- Cambio interactivo de opciones de configuración durante la reproducción del vuelo.
- Almacenamiento de capturas de pantalla.
- Herramienta de edición gráfica que permite al usuario generar gráficos a mano alzada sobre la pantalla del SD.
- Representación de la distancia entre dos aeronaves cualesquiera sobre la representación en 2D del vuelo.
- Carga de distintos mapas: Google Maps, Google Terrain y Google Hybrid, como así también cartas de navegación Geo-referenciadas.

En la Figura 1.1 se muestra como sería la interacción entre los dispositivos de registro en la aeronaves y la reproducción de los vuelos posteriormente en las estaciones de Debriefing.

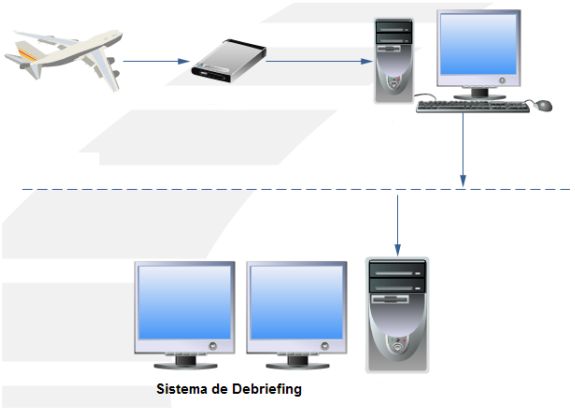

Figura 1.1: Sistema Registrador y Estaciones de Debriefing

Actualmente una estación (a modo de ejemplo) de Debriefing se compone de la siguiente manera (Figura 1.2):

Figura 1.2: Estación de Debriefing

La integración del sistema SRPV con el SD se presenta en la figura 1.3, donde puede observarse que el SD es claramente un subsistema de los diferentes módulos registradores de parámetros de vuelo o, en su defecto, SARM.

1.2. Problema de Investigación

En las visitas realizadas en carácter de reuniones participativas de varias brigadas aéreas pertenecientes a la FAA y en especial, a la EAM se realizó un relevamiento funcional en materia de adiestramiento y combate de los pilotos y al observar allí,

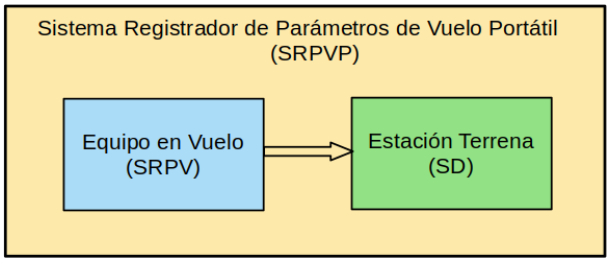

Figura 1.3: Arquitectura de Estación de Debriefing

la metodología o forma con la que los instructores de vuelo (post ejercicio de vuelo), realizan en salas específicas (denominadas Salas de Debriefing) de las instalaciones de cada brigada/escuela la devolución en materia de desempeño, conocimiento y aptitudes personales a cada piloto con respecto a la realización de ejercicios de adiestramiento previamente planificados dio lugar a la iniciativa o planteo inicial de un SD que permita al instructor proveer de una herramienta extra y versátil para la explicación más precisas de los ejercicios, maniobras, cruces, etc. Posterior a reiteradas visitas pertinentes, fue necesario plantearse (con el equipo de desarrollo) cómo modernizar y/o agilizar las reuniones post vuelo mediante la gestión de un SD y brindar así una metodología ágil, moderna que otorgue valor agregado a las brigadas y o escuelas de aprendices. De allí, surgió el planteo del equipo de desarrollo del centro. *¿Adquirir un Sistema de Debriefing Comercial o desarrollar uno nuevo y propio?* En conjunto con entidades de la Fuerza Aérea interesadas y con la DGID, se tomó la decisión de un desarrollo integral, adaptado a las necesidades específicas de los clientes y brindar allí el soporte y mantenimiento necesario continuo para mejoras del sistema. Cabe mencionar como dato adicional que este planteo de desarrollo de un nuevo SD para cubrir las necesidades de adiestramiento en combate a los pilotos de la FAA fue muy valorado por las brigadas y la EAM. En base a lo expuesto y después de un análisis riguroso, se confirmó la implementación de un nuevo SD, a desarrollar por personal capacitado de I+D, en el marco-convenio del proyecto existente denominado (como parte de un subsistema del mismo) FAS de la Fuerza Aérea Argentina y denominado *"SRPVP FAS D-AV 0180"* para utilizarse en las instituciones con la necesidad, entre otros, de control del tráfico aéreo de la aeronaves de la FAA montadas tanto con dispositivos SRPV como así también los vuelos

comerciales que utilizan la tecnología ADS-B. El objetivo concreto es la visualización en vuelo y post vuelo, todo en un mismo SD, de los diversos sistemas de navegación que intervienen en el espacio aéreo.

1.3. Justificación

El principal objetivo del SD es permitir tanto a los pilotos como a los instructores, la evaluación del desempeño y el nivel de adiestramiento alcanzado durante las distintas operaciones aéreas y/o vuelos de entrenamiento así como de las restricciones de las aeronaves. Además, permitirá ampliar las capacidades post vuelo de los SARM que no dispongan de la posibilidad de almacenamiento de datos (o se encuentren fuera de servicio por algún motivo) y que utilicen el SRPVP. El diseño y desarrollo del sistema al ser desarrollado *in-house* por la FAA permitirá a la propia Fuerza proponer mejoras a partir de nuevas funcionalidades que faciliten el trabajo de adiestramiento en combate (mantenimiento y soporte técnico). Además es de destacar que este proyecto se realizará con un presupuesto mínimo, comparado con productos comerciales similares (*ForeFlight*[1] para citar un ejemplo) a gran escala y muy completos pero con presupuestos más elevados (mayor de U$S5000 dolares estadounidenses según publicación del sitio y que requieren la adquisición de aplicaciones por separado sumado también, el costo de mantenimiento y soporte técnico), por otro lado, su funcionamiento es completamente *"off-line"* es decir, es completamente autónomo e independiente a una conexión de internet, esto está propuesto así porque los ejercicios y/o misiones que llevan a cabo los pilotos de combate están dentro de la jurisdicción de la FAA y son de carácter estrictamente confidencial. A diferencia de otros sistemas, como ForeFlight para su utilización, es mandatorio (obligatorio) la *nube* para poder llevar a cabo el Debriefing de forma correcta, completa y satisfactoria. Este sistema actualmente está en su versión preliminar y se utiliza por los diferentes destinos y brigadas de la FAA; cabe mencionar, que en la actualidad está en proceso de homologación (no finalizado aún al término de la redacción de este trabajo).

[1] https://foreflight.com/products/foreflight-web/

1.4. Antecedentes

Este proyecto surge, en el año 2010, ante la necesidad de la IV Brigada Aérea ubicada en la provincia de Mendoza, la que solicita oportuna y formalmente un SD, el cual debería específicamente tomar datos reales almacenados en un cartucho de una aeronave IA-63 Pampa II y, que a partir de allí, realice la reproducción de los diversos ejercicios de adiestramiento y combate de forma completamente autónoma (sin necesidad de acceso a la nube) en un plano de representación gráfica en 2D y 3D además de características adicionales como ser: gráficas en pantalla, generador de gráficos, distancias entre aeronaves, mensajería de eventos tanto de aeronaves como así de el o los vuelos realizados, entre otros. Este sistema fue desarrollado íntegramente e instalado para finales del año 2012 en la IV Brigada Aérea. A principios del año 2019, la EAM en conjunto con las IV y V Brigada Aérea, solicitan en conjunto con el Sistema SRPV un SD que pudiera no solo integrar el SRPV y el SARM IA-63 Pampa II/II, sino además otros SARM como A4-AR y ADS-B. Para ello, se realizó un análisis exhaustivo de la conveniencia de realizar un SD completamente nuevo o por el contrario, adquirir uno con licencia comercial multi-usuario. Como resultado de este análisis y teniendo como *"know how"* el logrado en el desarrollo del SD en la IV Brigada Aérea, se decidió (de forma conjunta y mancomunada por el equipo del proyecto) el desarrollo de un SD completamente nuevo, pero con la utilización de la experiencia, análisis y diseño del Sistema anterior. Posterior a ello, a finales del año 2020, se presentó un versión preliminar (prototipo) del SD desarrollado con funcionalidades básicas para ser evaluado como el primer corte *RC* por los usuarios. En febrero de 2022, se presentó la versión completa del SD, que comprende la integración conjunta de los siguientes sistemas: SRPV, ADS-B, SARM A4-AR, SARM IA-63 Pampa II/III y GPS IA-58 Pucará. En la actualidad, y desde junio de 2022, el SD esta en proceso de homologación, en un trabajo conjunto entre el desarrollador y la DEyH (IEEE 830, 2008).

1.5. Objetivos

Los requerimientos específicos del proyecto se generan a través de un objetivo general que esta enmarcado dentro del RO del cual surge el proyecto general y a lo que se pretende llegar. Este se desglosa en objetivos más específicos (contrastados con los requerimientos) ordenados cronológicamente y así queda establecida la trazabilidad para satisfacer el objetivo general.

1.5.1. Objetivo General

Como se introdujo en la sección anterior, el Objetivo General es el desarrollo de SW funcional, siguiendo la metodología del Proceso Unificado, denominado SD que realice, principalmente, la visualización en planos 2D y 3D, 100 % autónomo de conexión a Internet y a instalarse en salas específicamente diseñadas y destinadas al Debriefing ubicadas en cada destino de la Fuerza Aérea Argentina (FAA). Este objetivo general imparte en objetivos más específicos que están ligados unívocamente a los requerimientos técnicos del SD.

1.5.2. Objetivos Específicos

- Introducción e investigación tecnológica de herramientas para la visualización en 2D de mapas en tiempo real y objetos 3D.
- Adquisición de librerías para manejar mapas y objetos 3D y comparar el funcionamiento con licencias privadas (*benchmarks*).
- Análisis de componentes de SW (librerías entre otros) y factibilidad de funcionamiento en situaciones extremas (Test de Robustez).
- Desarrollo de documentos que contemplen artefactos de análisis y diseño con el objetivo de definir las funcionalidades técnicas e interfaces a nivel de SW.
- Análisis de factibilidad de carga de cartografía geo-referenciada.
- Análisis de factibilidad de desarrollo de modelos 3D a escala de los diferentes SARM y objetos (por ejemplo: SRPV).

- Prueba y ensayo del funcionamiento de unidades de SW a su versión preliminar, análisis del comportamiento de los componentes visuales en sincronización con los datos de entrada.
- Generación de informes técnicos sobre los ensayos y pruebas realizadas contrastados con los requerimientos técnicos propuestos, estos normalmente llamados procesos de VyV.

1.6. Aspectos Teóricos

Para el desarrollo de este sistema y el seguimiento del ciclo de vida del SD, se emplearon técnicas metodológicas que provienen de la Ingeniería del Software, más específicamente, el Proceso Unificado de Software (Ivar Jacobson y Rumbaugh, 1999), que provee los lineamientos necesarios centrados en, la arquitectura, enfocado en los riesgos, que presta cualidades tales como *"Iterativo e Incremental"*. La clasificación que plantea el Proceso Unificado en etapas de desarrollo es la siguiente (Ver Figura 1.4):

- Captura de Casos de Uso.
- Requerimientos Técnicos.
- Análisis.
- Diseño.
- Implementación.
- Testing.

1.7. Aspectos Metodológicos

Al abordar la realización del desarrollo del SD, se plantearon y realizaron los siguientes procesos conforme a los lineamientos planteados en el punto anterior (Aspectos Teóricos) y ordenados de la siguiente manera conforme a la metodología que impone el Proceso Unificado[2].

[2]El modelo de Despliegue no se aplicará en este desarrollo.

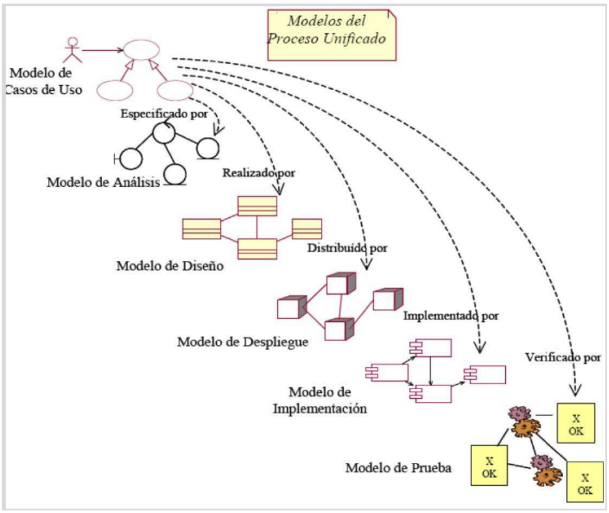

Figura 1.4: Etapas del Proceso Unificado

1.7.1. Captura de Casos de Uso

Se realizó el proceso que consiste en la captura de requerimientos en artefactos denominados Casos de Uso, conjuntamente con el RO oportunamente confeccionado por los clientes con el objetivo de comenzar con el análisis y validación de los requerimientos funcionales y no funcionales obtenidos de los casos de uso especificados y corroborar su trazabilidad. Este punto habilita al desglose de estos requerimientos en los formalismos de análisis.

1.7.2. Requerimientos Técnicos

Como se mencionó en la sección anterior, los requerimientos funcionales o técnicos desglosan los casos de uso en artefactos de clases de análisis en donde permiten especificar funciones especificas del SD, por citar un ejemplo, "la reproducción de un ejercicio se realiza pulsando un botón denominado Play", esto indica que tal acción esta unívocamente asociado al caso de uso que determina la reproducción con las clases de análisis que llevan a cabo esa tarea, hablando para esta instancia a un nivel superior (sin especificar aún el detalle del diseño e implementación).

1.7.3. Análisis

En esta etapa, se realiza el proceso de desglose de los requerimientos técnicos impartidos de los casos de uso en sus correspondientes artefactos de clases de análisis. En este punto, es en donde se definen las interfaces internas y externas del SD, los componentes de control y las clases de análisis que albergan datos del SD. Estos procesos deben estar completamente alineados a los procesos de captura de casos de uso.

1.7.4. Diseño

La etapa de diseño establece los lineamientos necesarios para desglosar los artefactos de clases de análisis en artefactos de clases de diseño. Estas clases, ya bien definidas y validadas con sus métodos y atributos que serán parte del código fuente (implementación) aplican también si es requerido, como *buenas prácticas de programación* (IEEE/EIA 12207, 2017), los patrones de diseño y jerarquías entre ellas (como asociación, herencia, composición, agregación, etc) dependiendo del diseño impartido de las clases de análisis y en todo momento, se corresponde la trazabilidad a los artefactos de análisis que a su vez, estos provienen de los requerimientos originados por los casos de uso.

1.7.5. Implementación

La implementación se lleva a cabo partiendo de los artefactos de diseño (clases de diseño) y contrastado a las funcionalidades a medida que avanza el desarrollo con los casos de test unitarios y realizando las correcciones pertinentes del SD. Posterior a ello, ya teniendo una versión preliminar del SD (versión Beta), y habiendo concluido los test unitarios de componentes de software se debe ejecutar los test de integración contemplados en las solicitudes de los requerimientos y verificando los resultados de la ejecución de dichos test a la integración final del SD y de esta manera, llegar a su versión RC.

1.7.6. Testing

En el momento de la especificación de los requerimientos funcionales y no funcionales, se realiza los procesos de validación y verificación que a partir de ellos, mediante la generación de Casos de Test conformando una *Test Suite* y posteriormente Test de Integración debido que estos procesos tienen que estar unívocamente relacionados a los procesos de requerimientos.

1.7.7. Cronograma

A continuación, se presenta el cronograma de forma resumida para el SD donde se muestran los avances **incrementales** del mismo. Los avances **iterativos** no se muestran por cuestiones de espacio del presente documento, debido a que son numerosos.

Actividad	Meses												Producto Entregable	Resultado Esperado	Entidad Ejecutora
	2	4	6	8	10	12	14	16	18	20	22	24			
Actividad 1: Especificación de Casos de Uso.													Documento de Casos de Uso. Documentos de Ensayos. (Validación y Verificación)	Tener validados los casos de uso contrastados con el requerimiento general y contrastados con los ensayos a ser realizados al sistema.	CITeA
Actividad 2: Especificación de Requerimientos Funcionales y no Funcionales a partir de los casos de uso.													Documento de Requerimientos del Sistema a Desarrollar.	Obtener especificaciones del Sistema a desarrollar en base a los Casos de Uso relevados en la etapa anterior.	CITeA
Actividad 3: Análisis y Diseño de la Arquitectura del SD.													Documento de Diseño del y Arquitectura del SD.	Obtener los diagramas de componentes de diseño y arquitecturas con sus respectivas interfaces.	CITeA
Actividad 4: Desarrollo e Implementación de la Arquitectura del SD.													Desarrollo de SW consensuando en el documento de diseño estipulado anteriormente.	Armado de ambiente en PC para probar el corte de SW implementado en su versión Beta.	CITeA
Actividad 5: Realización de ensayos y Testing del SD.													Documento de diseño de Ensayos Documento de Reporte de los ensayos de integración.	Poder realizar todas las etapas de pruebas necesarias del sistema antes de etiquetado como una RC (Release Candidate)	CITeA
Actividad 6: Evaluación y Homologación del SD.													Documento de Reporte de Ensayos de validación.	Ensayar el Sistema desarrollado conforme a las directivas de Validación y Verificación propuestas por la Dirección de Evaluación y Homologación.	DEyH
Actividad 7: Informe Final													Documentos de Informe Final de Proyecto.	Documentos finales del Proyecto necesarios para la obtención de su homologación.	CITeA

Tabla 1.1: Cronograma (tentativo) de Desarrollo del Sistema de Debriefing

1.7.8. Recursos Necesarios

1.7.8.1. Factibilidad Técnica

La metodología seleccionada para llevar a cabo el ciclo de vida[3], el Proceso Unificado, con sus fases bien definidas de inicio, elaboración, construcción y transición, proporciona una estructura robusta y flexible que se adapta a las necesidades específicas del proyecto. Este enfoque iterativo permite una gestión eficaz de los riesgos, mejorando la previsibilidad y la calidad del producto final. Además, el centro de desarrollo de SW que va a llevar adelante el proyecto cuenta con profesionales capacitados y herramientas avanzadas que facilitan la implementación de prácticas de ingeniería de SW, como el modelado visual, la gestión de requisitos y la integración continua. Estos factores, combinados con la naturaleza colaborativa del Proceso Unificado, aseguran que el proyecto pueda ser desarrollado de manera eficiente, con alta calidad y dentro de los plazos establecidos.

1.7.8.2. Factibilidad Operativa

Para el desarrollo del presente proyecto en el ámbito laboral (centro I+D perteneciente a la DGID) cuenta con los insumos necesarios como ser PC de escritorio (entre otros) en condiciones operativas para el desarrollo del SD, además de contar con profesionales de la misma índole (desarrolladores de SW), que permite diagramar coordinadamente y colaborativa-mente un equipo de trabajo para desarrollar las distintas etapas que conlleva el proyecto.

1.7.8.3. Factibilidad Económica

Se requiere incurrir en gastos de adquisición de licencias de librerías específicas (en caso de ser necesario) para llevar a cabo la visualización del sistema y de licencias del entorno de programación escogido para implementar la solución. Estos costos se detallan a continuación (cabe aclarar son valores aproximados y pueden variar dependiendo de cotizaciones al día)

[3]https://blog.ganttpro.com/es/ciclo-vida-de-un-proyecto/

Cantidad	Descripción	Costo Total
1	Licencia Full Multi-Usuario de librería 3D	$ 70.000
1	Licencia Full Multi-Usuario de entorno de (IDE) programación de Microsoft Visual Studio.	$ 90.000
1	PC de escritorio con Windows 10 para realizar los Ensayos.	$ 120.000
1	Notebook con Windows 10 para realizar la puesta en marcha de la versión RC del SD.	$ 110.000
	Total	$ 390.000

Tabla 1.2: Presupuesto presentado de Software (SW)

Capítulo 2

Desarrollo del Sistema de Debriefing

2.1. Introducción

En términos generales una misión o ejercicio de adiestramiento de vuelo en una brigada operativa consta de tres etapas: un explicación técnica de lo que se quiere o pretende realizar denominado *briefing*, en donde se detalla por un instructor de vuelo a los alumnos en que constará el ejercicio o misión, la ejecución aeronáutica del ejercicio o misión propiamente dicha y supervisada por el instructor de vuelo y una evaluación posterior que en términos militares que se denomina *debriefing*.

En ambos casos, el briefing y el debriefing deben ocupar idealmente el mismo tiempo que el vuelo en ejercicio realizado, y en ambos se deben abordar todas las etapas y fases del vuelo. Durante el briefing se destacan los objetivos que los alumnos deben cumplir, mientras que en el debriefing se enfatizan el análisis, debates, conclusiones y lecciones aprendidas por parte del instructor a los alumnos de acuerdo a lo transcurrido en la misión/ejercicio. El SD se implementa en aulas específicamente ambientadas al briefing/debriefing del Escuadrón II CEPAC, en este caso quien fue el usuario que solicito el requerimiento principal de contar con un SW operativo para realizar puntualmente las sesiones de debriefing.

Como agregado extra a las funcionalidades descriptas, el SD contará con la posibilidad de marcar uno o más puntos de la reproducción para llevar a cabo un

análisis posterior o bien existirá la posibilidad de acceder a un evento determinado (lanzamiento de un proyectil por ejemplo). Para describir el desarrollo del SD en su completitud, se comenzó con una descripción más detallada de cada etapa del desarrollo aplicando del Ciclo de Vida introducido en la sección 1.7.

2.2. Requerimientos Técnicos

2.2.1. Visión de los requerimientos de sistema

En base al apartado anterior se detallan más técnicamente cada grupo de requerimientos.

2.2.1.1. Generar Vuelo

Este requerimiento es la etapa final del vuelo que se pretende generar en donde las condiciones iniciales del GV (cantidad de aeronaves, especificaciones de las mismas como tipo, color, etc. datos de entrada de los vuelos, fecha, etc) ya están cumplidas satisfactoriamente y de esta manera se generará un archivo (en la ubicación que el usuario especifique) con el formato correcto correspondiente a ser interpretado por el SD.

2.2.1.2. Seleccionar fecha

En este punto, se debe seleccionar la fecha en la que fue realizado el vuelo, independientemente que sea un vuelo de una aeronave en particular o una misión conjunta de varios vuelos de diferentes aeronaves, todos ellos deben reglamentariamente coincidir con la fecha seleccionada. De esta manera se permiten generar ejercicios combinados y realizados en fecha actual o en fechas históricas.

2.2.1.3. Seleccionar vuelos disponibles

Una vez que se carga la grilla de vuelos (con los correspondientes datos de la aeronave, piloto, copiloto, etc) a ser generados de las diferentes entradas, el usuario

debe seleccionar los vuelos que desea generar. Una vez cumplimentada esta tarea, se esta en condiciones de general el vuelo a ser reproducido por el SD.

2.2.1.4. Seleccionar hasta 20 vuelos cargados

En base a la selección anterior por parte del usuario, solo sera posible seleccionar una cantidad de hasta 20 aeronaves en conjunto como máximo, independientemente que sean del mismo tipo como de diferentes tipos. Caso contrario, es decir, que se supere esa cantidad el GV, deberá informar con un mensaje emergente al usuario que ha superado el cupo máximo permitido.

2.2.1.5. Graficar Parámetros

Se contará con una herramienta parte del SD que permitirá generar gráficos en ejes cartesianos en base a los datos de las diferentes aeronaves. Estos datos deben ser común a todas las aeronaves que se deseen graficar y como resultado se ofrecerá de forma visual un gráfico comparativo en ejes cartesianos en el cual un eje representará el tiempo y el otro eje representará los valores de las aeronaves identificados en diferentes colores para poder resaltar mejor la comparación.

2.2.1.6. Seleccionar Parámetros a Graficar

Para completar con el requerimiento anterior, es obligatorio seleccionar las aeronaves que intervienen y sus parámetros de vuelo (valores propias de las mismas) a ser representada visualmente para una posterior comparativa. Cabe aclarar que también puede ser realizado el gráfico de solo una aeronave. La selección dependerá de la decisión del usuario.

2.2.1.7. Realizar Debriefing

En las próximas secciones se presentaran los requerimientos propios del SD que deben satisfacerse para lograr el objetivo de visualización de parámetros de vuelos registrados de las diferentes aeronaves y SARM.

2.2.1.7.1. Seleccionar Vuelo: Una vez cargados los vuelos, deben seleccionarse aquellos que se deseen reproducir. Para ello, se selecciona la casilla correspondiente al vuelo y desde allí, se podrá cambiar parámetros como el tipo, color de la aeronave como así también los indicativos de piloto y copiloto. Una vez completada esa operación se deberá realizar el Debriefing.

2.2.1.7.2. Visualizar 2D: En la pantalla principal del SD, se visualizara un sector destinado a la visualización 2D, en donde se representarán los objetos (aeronaves) con su correspondiente identificación (indicativos de piloto, copiloto, color, etc) y sus parámetros de vuelo (Latitud, Longitud, Altitud, Actitud, Aceleraciones, etc) sobre un mapa que puede seleccionarse entre: Google Maps, Google Terrain y Google Hybrid sobre diferentes niveles de alejamiento y acercamiento (zoom) desde el nivel 4 hasta el nivel 15.

2.2.1.7.3. Visualizar 3D: Del mismo modo que la visualización 2D, el SD contará con un sector destinado a la visualización 3D, en donde se presentará el modelo 3D de la o las aeronaves a escala, con su correspondiente color (seleccionado previamente por el usuario en la carga del vuelo) sobre una superficie en modelo 3D que estará georeferenciadas y además con la exactitud del cálculo de altura basándose en los modelos de elevación DEM(extraídas desde la NGA (teams at U.S. military, 1990)) que contienen la altura de todo el territorio argentino, basándose en los datos de posición geográfica (Latitud y Longitud). De esta manera, se podrá apreciar las maniobras en altura, despegue y aterrizaje con su correspondiente referencia de altura en base a la altitud de la aeronave. Cabe destarar que se visualizará también la "estela" de la aeronave, es decir el trayecto que va realizando a medida que se desarrolla el o los vuelos.

2.2.1.7.4. Ajustar Parámetros: El usuario deberá realizar cambios a su criterio de los parámetros de vuelo a través de una solapa especial, en donde podrá elegir que parámetros visualizar, como así también la opción de "siempre permanente" o que se visualicen cuando se desliza el puntero del mouse sobre el objeto en estudio. De igual manera se podrá modificar el tamaño de la **"estela"** (trayecto sobre el

mapa) como así de igual modo, su ancho lógicamente a criterio del usuario.

2.2.1.7.5. Reproducir, Pausar y Detener: El usuario contará con control de reproducción con las acciones de comenzar (Play), pausar (Pause), Detener (Stop) la reproducción del o los vuelos. De esta manera, se podrá realizar los análisis pertinentes en todo momento, en maniobras, aterrizaje, despegue y demás que así se consideren. Poseerá la capacidad mediante este control de reproducción de repetir el o los vuelos tantas veces sea necesario si así lo considere el usuario.

2.3. Casos de Uso

2.3.1. Visión general

La aeronave que interviene registra una serie de datos durante el vuelo, los cuales son obtenidos y descargados para ser procesados por el SD. En base a estos datos, el SD permitirá simular el vuelo junto a la reproducción sincronizada del mismo. Partiendo del RO del SRPVP: *"Sistema Registrador de Parámetros de Vuelo portátil, para ser utilizado en un Sistema de Debriefing"* se definen los casos de uso generales del SD y posterior a ellos, los requerimientos técnicos específicos que se corresponderán a sus componentes de diseño.

2.3.1.1. Estructura de Casos de Uso General

Se presenta a continuación la captura de requerimientos modelado en casos de uso a partir del RO planteado anteriormente (Figura 2.1).

Este caso de uso, describe el comportamiento que deberá realizar el SD en base a la solicitud descripta previamente por el usuario. De allí, se desglosa en sub casos de uso que describen técnicamente cada una de las funciones para comprender y modelar las principales funciones del SD.

2.3.1.2. Estructura especifica de Casos

Como se mencionó en el punto anterior, el caso de uso general es expandido o desglosado en funciones más técnicas y especificas, dando inicio a los siguientes casos

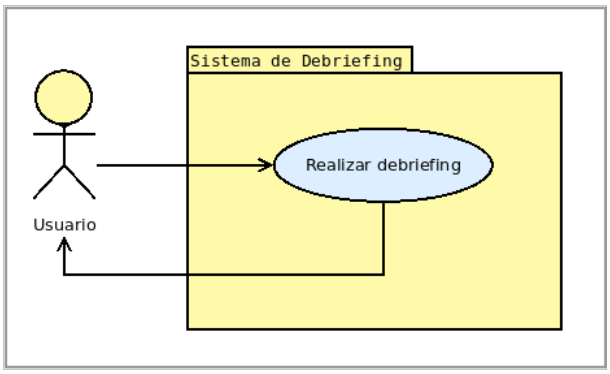

Figura 2.1: Diagrama de Casos de Uso del SD

de uso: el GV, el Generador de Gráficos y el SD propiamente dicho (Figura 2.2).

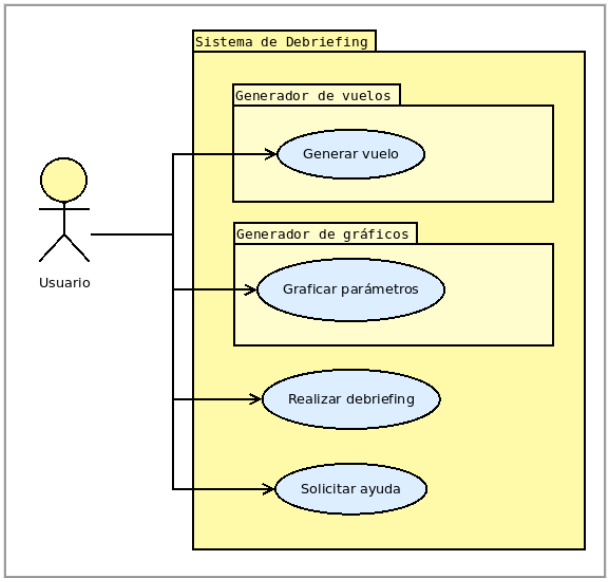

Figura 2.2: Diagrama de Casos especifico por funcionalidad

2.3.1.3. Generador de Vuelos

Analizando el caso de uso del GV, surgen los siguientes requerimientos:

- Generar Vuelo.
- Seleccionar fecha.
- Seleccionar vuelos disponibles.
- Seleccionar hasta 20 vuelos cargados.

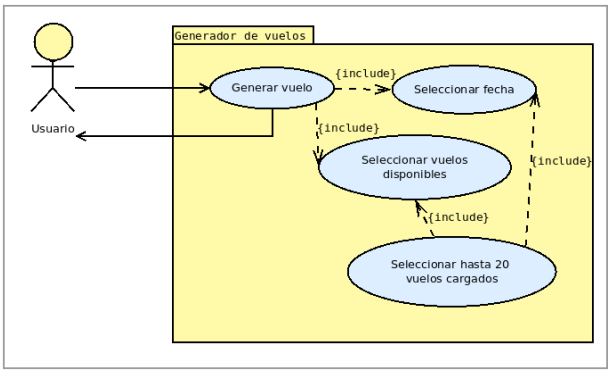

Figura 2.3: Diagrama de Requerimientos Funcionales

Estos requerimientos se corresponderán a los componentes (diseño) entablados en la arquitectura de diseño (Figura 2.3).

2.3.1.4. Graficar Parámetros

Correspondiente al caso de uso que especifica como se deben graficar los parámetros de vuelo, se presenta el diagrama con los siguientes requerimientos establecidos.

- Graficar Parámetros.
- Seleccionar Parámetros a Graficar.

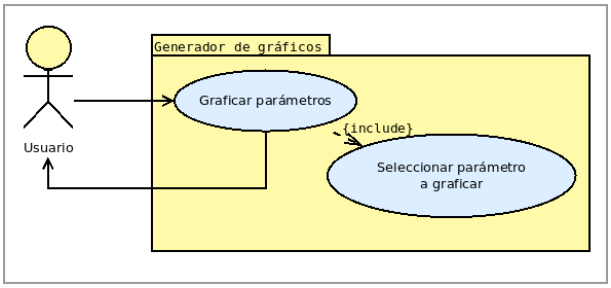

Figura 2.4: Diagrama de Requerimientos Funcionales (cont)

Estos dos requerimientos se corresponderán a los componentes (diseño) entablados en la arquitectura de diseño (Figura 2.4).

2.3.1.5. Realizar Debriefing

Correspondiente al caso de uso que especifica como se debe realizar la generación de información para la posterior visualización del vuelos a través de sus

correspondientes parámetros de vuelo, se presenta el diagrama con los siguientes requerimientos establecidos.

Realizar Debriefing:

- Seleccionar Vuelo.
- Visualizar 2D.
- Visualizar 3D.
- Ajustar Parámetros.
- Reproducir, Pausar y Detener.

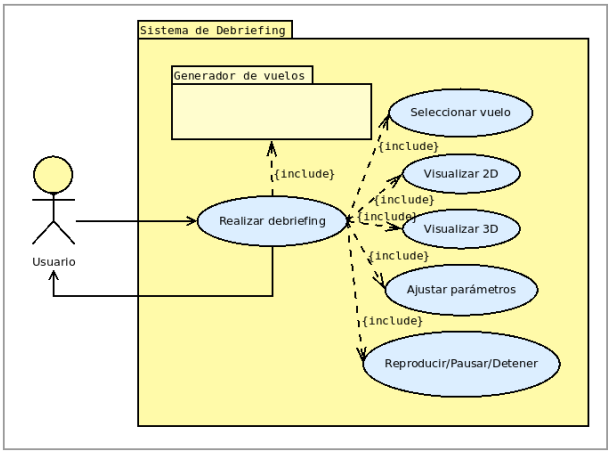

Figura 2.5: Diagrama de Requerimientos Funcionales (cont)

Estos requerimientos se corresponderán a los componentes (diseño) entablados en la arquitectura de diseño (Figura 2.5).

2.4. Análisis

A modo de ejemplo, la aeronave que interviene registra una serie de datos durante el vuelo, los cuales obtenidos y descargados para ser procesados por el SD (Figura 2.6).

En base a estos datos, el SD permitirá simular el vuelo junto a la reproducción sincronizada del mismo. A partir de los datos registrados durante el vuelo con una representación en 2D y 3D, simulada de estos. También se ofrecerá la posibilidad de

```
[General]
ACnumber = 1
A4-AR, AC1, Blue, C-918, TB01, 27-03-20, 08:23:54, 12:43:43, LOCAL, TB0, No Description...

[FlightData]
40417000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, -0.0003662109375, -0.000732421875, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40422000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, 0, 0.0006103515625, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40423000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, 0, -0.0013427734375, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40424000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, 0.0013427734375, -0.00048828125, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40425000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, -0.0018310546875, 0.0006103515625, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40426000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, 0.0015869140625, -0.0015869140625, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40426500, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, -0.0015869140625, 0.0010986328125, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40427000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, 0.0006103515625, 0, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40428000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, -0.000244140625, 0.0009765625, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40428500, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, -0.0001220703125, -0.0001220703125, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40429000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, 0.0008544921875, -0.0006103515625, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40430000, AC1, ACDAT, ACDAT, -33.7238332908601, -65.3753333073109, 0, 177, 0, 5, 175, 0.001953125, -0.0009765625, 0, 70, 50, 19, 1516, 0, 0, 0, 0
40431000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.0062255859375, 0.00390625, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40464000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, 0.00341796875, 0.001953125, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40464500, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.00390625, -0.0018310546875, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40465000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.0048828125, -0.003173828125, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40466000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, 0.004150390625, 0.0018310546875, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40466500, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.00439453125, 0, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40467000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, 0.0029296875, -0.00146484375, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40468000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, 0.001953125, -0.0064697265625, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40470000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.0020751953125, -0.003662109375, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40470500, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.0009765625, 0.000244140625, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40471000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.0093994140625, 0.0028076171875, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40472000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.0072021484375, 0.00390625, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40472500, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, 0.0052490234375, -0.0029296875, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40473000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.003662109375, 0.0037841796875, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40474000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, 0.0052490234375, 0.0015869140625, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40474500, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.0025634765625, -0.0018310546875, 0, 70, 50, 19, 1514, 0, 0, 0, 0
40475000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, 0.0023193359375, 0.004150390625, 0, 70, 50, 20, 1514, 0, 0, 0, 0
40476000, AC1, ACDAT, ACDAT, -33.7202265579253, -65.3775592055172, 0, 178, 0, 5, 175, -0.001220703125, -0.004150390625, 0, 70, 50, 20, 1514, 0, 0, 0, 0
```

Figura 2.6: Fragmento de datos de entrada

solicitar la indicación de la distancia real entre cada aeronave.

El SD presentará una interfase amigable (como se muestra en la figura a continuación), facilitando su uso para realizar la función de debriefing, esto es, la reproducción *off-line* de la misión de vuelo (Figura 2.7).

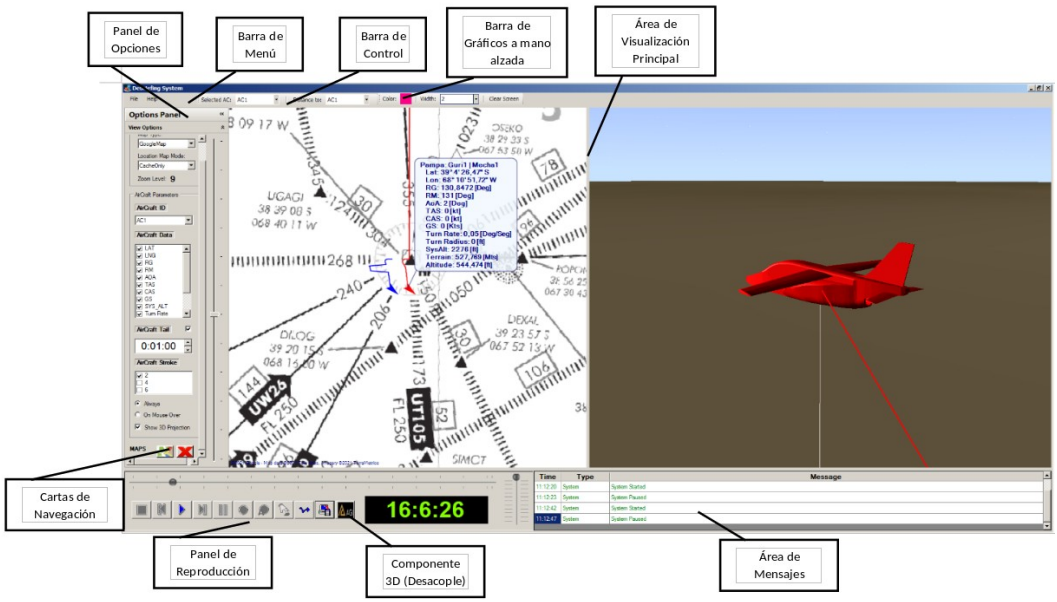

Figura 2.7: Entorno del Proyecto de Software

2.4.1. Arquitectura estática del software

Como se mencionó anteriormente, la actividad diaria de vuelo consta de una reunión previa (briefing), la actividad práctica propiamente dicha y la reunión posterior (debriefing). Tanto la previa como la posterior deben insumir un tiempo al menos igual al del vuelo realizado y en ambas se deben desarrollar todas las instancias y fases del mismo, haciendo en la previa, hincapié en los objetivos, y en la posterior, hincapié en los análisis, conclusiones y enseñanzas. Para ello, el ámbito de aplicación del SD serán las aulas de briefing/debriefing del Escuadrón correspondiente. Asimismo el SD ofrecerá capacidad para llevar a cabo ensayos de vuelos, la cuál será utilizada en el ámbito del CEV.

2.4.1.1. Estructura estática del software – Nivel 0 (Level 0)

A Continuación, se muestra el diagrama de Análisis principal del SD (Figura 2.8).

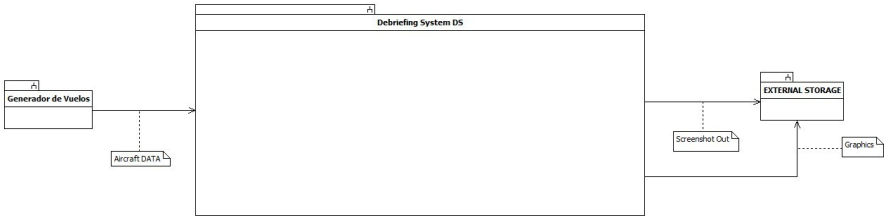

Figura 2.8: Diagrama de análisis del sistema – Nivel 0

En la figura, se observa que el SD posee una entrada y una salida, se ofrece una descripción de las mismas.

- Entrada: La entrada al SD consiste de un archivo en TXT. El archivo almacena todos los datos de actitud de vuelo de la aeronave en forma cíclica.
- Salida: Básicamente el SD ofrece una salida la cual es almacenada en el disco duro de la PC en donde se ejecuta el SD. Esta salida es capturas de pantallas en formato PNG, además de un archivo de tipo imagen que se corresponde a gráficos sobre los valores de algún datos deseado (por ejemplo: Latitud) del SD.

2.4.1.2. Estructura estática del software – Nivel 1 (Level 1)

Este gráfico representa a gran escala los componentes internos compuestos del SD y las interfaces correspondientes que especifican el flujo de datos o información intercambiada entre dichos componentes (Figura 2.9).

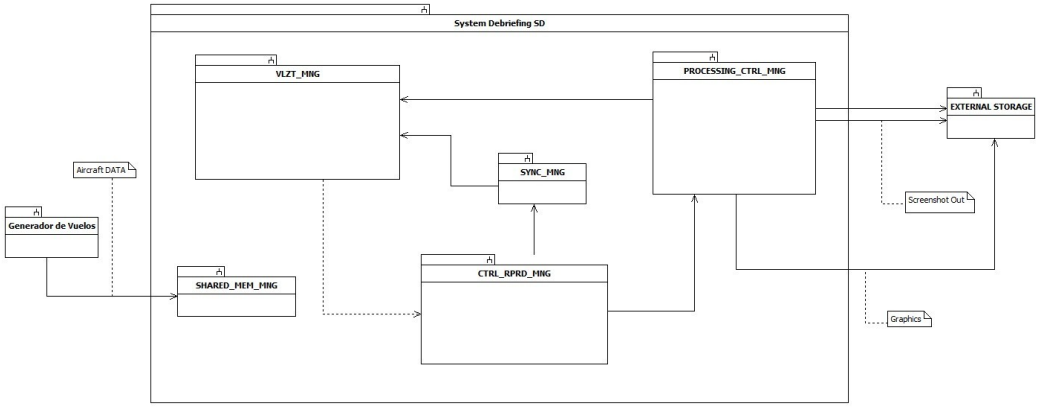

Figura 2.9: Diagrama de diseño – Nivel 1

- Componente VLZT MNG: Este componente es el encargado de administrar la visualización del SD sin llevar a cabo ningún tipo de procesamiento de datos, es decir, no tiene lógica de control, es meramente visual.
- Componente CTRLR PRD MNG: Este componente es el encargado de controlar la reproducción del SD. Además, el componente proporciona la posibilidad de generar y administrar marcas en la sesión con el objetivo de tener un acceso rápido a las mismas. También se encarga de gestionar los eventos registrados durante el vuelo de la aeronave. Este componente proporciona datos al componente SYNC MNG que le permitan a este último gestionar la sincronización del SD.
- Componente PROCESSING CTRL MNG: Este componente se encarga de llevar a cabo todos los cálculos necesarios durante la sesión de debriefing. Esto quiere decir que este componente agrupa toda la lógica de control dentro del SD. El componente toma datos de la memoria compartida los procesa y ofrece salida que alimenta al componente VLZT MNG y el archivo de captura de imágenes.

- Componente SYNC MNG: Este componente es el encargado de administrar la sincronización de todos los datos disponibles para el debriefing. Como entrada, este componente recibe comandos desde el componente CTRL RPRD MNG tales como play, pause, stop, introducir/saltar a una marca o bien saltar a un evento específico. En cuanto a la salida, el componente alimenta al componente VLZT MNG indicándole los datos que debe tomar de la memoria compartida para mostrar por pantalla.
- Componente INPUT SHARED MEM MNG: Este componente es el encargado de almacenar los datos de entrada en la memoria compartida y gestionar dicha memoria. Como entrada, este componente lee el archivo proveniente del GV que corresponde a un repositorio configurable. Una vez leídos estos datos, se encarga de almacenarlos de forma adecuada en la memoria compartida. Como salida, se encarga de alimentar al resto de los componentes cuando estos lo requieran.

2.4.2. Arquitectura dinámica del software

El SD se implementa sobre una arquitectura hardware x86 de 64bits con mínimo 8Gb de RAM y 300GB de espacio libre de HD. El sistema operativo necesario será Windows 7 (64Bits) profesional SP2 en su versión en español o Windows 10 (64Bits) en su versión en español.

2.4.2.1. Contexto de interfaces

El SD, como producto final, se ubica en este esquema el cual interactúa con otros sistemas (Figura 2.10).

En particular, el SD define la interfaz a través de la Interfase del GV. Este software se encarga de generar la totalidad de los datos provenientes de las diferentes entradas que pueden ser por ejemplo: un disco digital, ya sea CD, DVD, unidad Flash USB, DTM o wifi. Finalmente, los datos son utilizados por el SD para llevar a cabo la reproducción del vuelo.

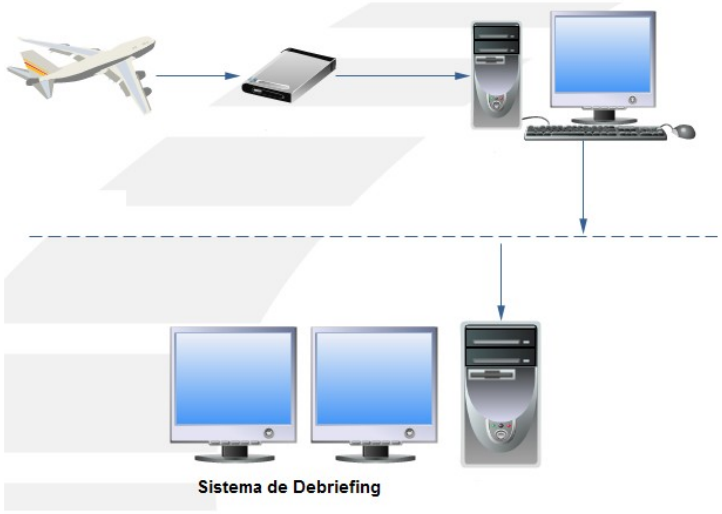

Figura 2.10: Interfaces Externas

2.4.2.2. Estimación de memoria y tiempo de CPU

En el contexto del SD operando a la totalidad de su capacidad, esto es, utilizando la capacidad completa de visualización, el SD no superará en ninguno de los modos de operación, en un porcentaje del 50 de uso de memoria y procesador. En el contexto del SD en estado ocioso, se mantendrá en todos los casos un uso máximo entre un 5 % y 10 % de memoria y procesador. Es importante remarcar que ninguna de estas estimaciones tiene en cuenta el consumo de recursos propio del sistema operativo y componentes que se encuentren en ejecución en el host del SD.

2.4.2.3. Estándares de diseño, convenciones y procedimientos

En la arquitectura de análisis del SD se utilizó una notación basada en UML (Rumbaugh y Booch, 2004) para la realización de los diagramas y la documentación. El desarrollo del SD se llevo a cabo utilizando las metodologías orientadas a objetos y se tuvo en cuenta las consideraciones presentadas a continuación. Cabe como aclaración que la metodología utilizada como guía para el desarrollo del SD (Proceso Unificado (Ivar Jacobson y Rumbaugh, 1999) más la utilización de lenguaje especial denominado UML) se utilizan algunas de las etapas (no todas) de la metodología de desarrollo mencionada. Esto es, porque esta metodología no solo abarca al desarrollo de pequeños sistemas, sino también de grandes sistemas de software

cuyas magnitudes requieren grandes grupos de personas repartidas por diferentes puntos geográficos hasta incluso países de allí, su alta complejidad. Para este caso de desarrollo mas pequeño solo se utilizaron las etapas de desarrollo mandatorias por así decirlo, dejando de lado áreas que no aplican a este fin.

Modularidad	El sistema se encuentra modelado a técnica de programación orientada a objeto en la cual, es decir, para cada componente se definen los roles o responsabilidades funcionales intercambiando mensajería con otros componentes sobre interfaces correctamente representadas. De esta manera, colaborativa mente, los componentes conforman la funcionalidad completa del sistema.
Uniformidad	Decisión de diseño de basar el hardware del sistema en sistemas compatibles x86. Desarrollo del código fuente en lenguajes de alto nivel (C, C++ y visual C# express). Uso del sistema operativo Windows 7 y Windows 10.
Facilidad de inicialización	El sistema ofrece un formulario para la carga inicial de datos, por ejemplo, datos de la misión, aeronaves, pilotos, etc. de forma tal que la configuración se lleva a cabo de manera automática al usuario.
Facilidad de modificación del software	Metodologías de diseño Orientadas a Objetos. Principios de diseño del menor privilegio, encapsula miento, herencia y modularidad del software. Encapsula miento de información.
Facilidad de auditoría.	Uso del módulo *Build In Test* para la comprobación del correcto funcionamiento del resto de los módulos del sistema, conformando así, el estado de salud (*SOH*) del mismo.
Facilidad de debugging	El sistema ofrece un mecanismo de auditoría mediante mensajes de Log que permite detectar y filtrar anomalías, alertas y *"status ok"*. Además es posible detectar el nivel y la graduación en las que ocurrieron fallas posibles.
Diseño de componentes orientado a objetos	El diseño de componentes elaborado a través de entidades o componentes que representan a un conjunto de objetos e instancias de clases.
Soporte para futuras extensiones	Debido a la característica modular que posee el sistema y una definición clara de las interfaces garantiza que la incorporación de futuros componentes se realice de forma simple y ordenada.

Tabla 2.1: Estándares de Software

A continuación se presenta la etapa de diseño del software del SD, el cual contempla una jerarquización más profunda del SD en donde se muestran componentes de diseño, las interfaces entre ellos, que por un lado se traslada *"hacia abajo"* a la etapa de implementación y *"hacia arriba"* a la arquitectura de artefactos de análisis.

2.5. Diseño

2.5.1. Visión general de la arquitectura de diseño

La arquitectura del diseño del SD se puede observar gráficamente en la siguiente figura, que describe los módulos principales en los que se divide (Figura 2.11):

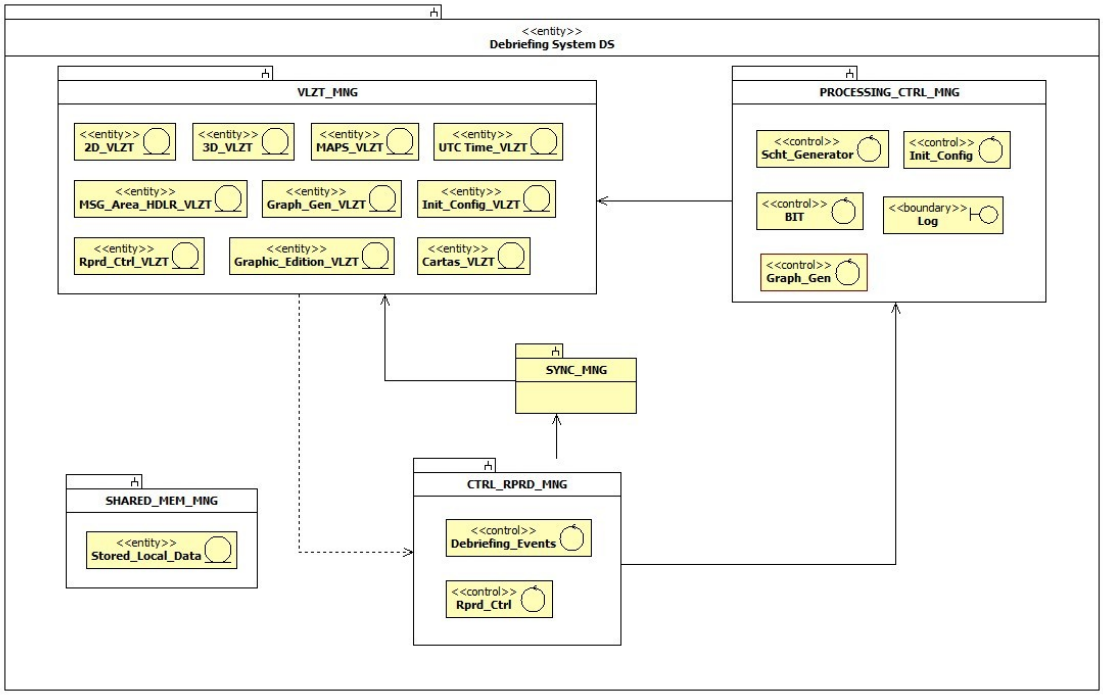

Figura 2.11: Diagrama de diseño del sistema – Nivel 2

A continuación se describe las responsabilidades de cada uno de los componentes y las relaciones entre los mismos.

2.5.1.1. VLZ MNG

El paquete VLZ MNG se encuentra integrado por diez componentes. A continuación se describirán componentes que constituyen paquete VLZ MNG.

2.5.1.1.1. 2D VLZT: El componente 2D VLZT es el encargado de proporcionar una vista en planta del terreno sobre el cual se encuentra volando la aeronave junto con un icono representativo de la misma. Además, el componente ofrece la posibilidad de mostrar la distancia entre dos o más aeronaves (seleccionable por el

usuario) y hasta catorce parámetros de interés dentro de un cuadro de texto acompañando al icono de cada aeronave en cuestión. También, el componente ofrece la posibilidad de llevar a cabo un dibujo sobre el área de reproducción utilizando el puntero del mouse, permitiendo al usuario remarcar cualquier detalle que considere de relevancia. El componente Aircraft DST se encarga de calcular la distancia entre todas las aeronaves utilizando los datos almacenados en la memoria compartida (INPUT SHARED MEM MNG). Luego, esta distancia es mostrada en los componentes 2D VLZT y/o 3D VLZT según el usuario lo requiera.

2.5.1.1.2. 3D VLZT: El componente 3D VLZT ofrece una vista en tres dimensiones del terreno junto con un modelo representativo de la/s aeronave/s en cuestión. De forma similar al componente 2D VLZT, en este caso también se ofrece la posibilidad (configurable por el usuario) de mostrar la distancia entre dos o más aeronaves y hasta catorce parámetros de interés dentro de un cuadro de texto que acompaña a cada aeronave. También, el componente ofrece la posibilidad de llevar a cabo un dibujo sobre el área de reproducción utilizando el puntero del mouse, permitiendo al usuario remarcar cualquier detalle que considere de relevancia. Este componente, también posee la lógica necesaria para calcular *"online"* en la reproducción los distintos niveles de elevación del terreno a través de Modelos de Elevación de terreno DEM, esta información es utilizada para calcular la altura que posee la aeronave en cuestión con respecto al terreno en todo momento en la reproducción del vuelo. El componente Aircraft DST se encarga de calcular la distancia entre todas las aeronaves utilizando los datos almacenados en la memoria compartida (INPUT SHARED MEM MNG). Luego, esta distancia es mostrada en los componentes 2D VLZT y/o 3D VLZT según el usuario lo requiera.

2.5.1.1.3. UTC Time VLZT: Este componente básicamente se encarga de mostrar por pantalla (en todo momento) la hora UTC correspondiente al vuelo (uno, dos o más aeronaves) para el cual se está realizando el debriefing.

2.5.1.1.4. MSG Area HDLR VLZT: El componente MSGAreaHDLRVLZT se encarga de imprimir todos los mensajes en el área de mensajes del SD. Básica-

mente no es posible encontrar con dos grandes grupos de mensajes, uno de estos está formado por mensajes de eventos, referentes al vuelo de cada aeronave y el otro constituido por los mensajes del SD. Los mensajes se muestran por pantalla en forma cronológica y distinguida por colores según el tipo.

2.5.1.1.5. Graph Gen VLZT: Este componente trabaja en conjunto con el componente Graph Gen. Este último es el encargado de generar gráficos de parámetros seleccionables por el usuario versus el tiempo, tales como Latitud, Longitud, Altitud, etc, mientras que el componente Graph Gen VLZT se encarga de llevar a cabo la visualización de los datos y acomodar los gráficos en la región de la pantalla seleccionada por el usuario.

2.5.1.1.6. Init Config VLZT: El componente Init Config VLZT se encarga de proporcionar un formulario de carga de datos al usuario para llevar a cabo la configuración inicial del SD. Este formulario ofrece la información general de la misión, permite seleccionar el número de aeronaves con las que se desea llevar a cabo el debriefing y el color que se desea asignar a cada una.

2.5.1.1.7. Graphic Edition VLZT: Este componente se encarga de brindar la posibilidad de realizar gráficos utilizando el puntero del mouse sobre los componentes 2D VLZT y 3D VLZT.

2.5.1.1.8. Rprd Ctrl VLZT: Este componente se encarga de manejar la visualización de los controles de reproducción. Detecta cuando el usuario lleva a cabo una acción, la muestra por pantalla y le comunica dicha acción al componente Rprd Ctrl, el cual posee toda la lógica de control de reproducción.

2.5.1.1.9. MAPS VLZT: Este componente permite cargar visualmente al componente 2D VLZT los tres diferentes tipos de mapas: Google Maps, Google Hybrid y Google Terrain, combinados con los niveles de zoom en el mapa que varia desde la escala 4 a la escala 15 de zoom. Estos mapas, están localmente (en cache) en la PC host para evitar el acceso a Internet a fin de acceder a ellos, pero de todos modos,

el usuario puede elegir de utilizar en combinación, esto es cache e Internet o solo Internet para los tres tipos de mapas.

2.5.1.1.10. Cartas VLZT: Esta funcionalidad esta asociada dentro del componente 2D VLZT y consiste en cargar sobre el mapa actualmente seleccionado, cartografía previamente digitalizada y georeferenciada, y de esta manera visualizar el objeto 2D sobre la misma con el objetivo de enriquecer el ejercicio sobre el vuelo seleccionado. Esto permite verificar el vuelo realizado mediante la utilización de cartas estandarizadas o mapas específicos personalizados de acuerdo al propósito del Debriefing.

2.5.1.2. PROCESSING CTRL MNG

El componente PROCESSING CTRL MNG es el encargado de llevar a cabo el procesamiento de datos en el SD. Este módulo está formado por componentes, los cuales se describen a continuación:

2.5.1.2.1. Graph Gen: El componente Graph Gen se encarga procesar los datos de entrada para que el componente Graph Gen VLZT muestre gráficos por pantalla de parámetros configurables por el usuario tales como Latitud, Longitud, Altitud, etc versus el tiempo.

2.5.1.2.2. Scht Generator: El componente Scht Generator se encarga de generar una captura de pantalla y almacenarla en formato PNG en el disco duro del SD. El directorio de almacenamiento es seleccionable por el usuario.

2.5.1.2.3. Init Config: Este componente se encarga de leer datos de la memoria compartida Data Prd y toma datos del componente Config File. Luego, proporciona estos datos al componente Init Config VLZT del paquete VLZT MNG el cuál se encarga de generar el formulario de inicialización y carga de datos del SD. Además, el componente se encarga de verificar si las aeronaves que se cargaron para realizar un Debriefing han volado juntas o no. En caso de haber volado juntas, el componente se lo comunicará al componente Init Config VLZT para que este permita la selección de

ambas aeronaves. Caso contrario, no se permitirá el Debriefing de ambas aeronaves en forma simultánea.

2.5.1.2.4. Log: El componente log se encarga de recibir todos los eventos (de relevancia) provenientes del resto de los componentes del SD. Cada evento, está formado por el mensaje correspondiente, el TOD, el nivel jerárquico y el tipo. Este registro de eventos se almacena en un archivo, que puede ser consultado externamente para diagnóstico de errores y tareas de mantenimiento. Además el componente se encargará de almacenar los mensajes que se registran en el componente de área de mensajes del SD.

2.5.1.2.5. BIT: Una vez finalizada la inicialización del SD, el componente BIT se encarga de testear la inicialización del resto de los componentes de software que constituyen el SD. Una vez finalizado el BIT, eleva un reporte al componente Log sobre el estado general del SD.

2.5.1.3. CTRL RPRD MNG

El paquete CTRL RPRD MNG contiene los componentes necesarios para el control y gestión del Debriefing. Maneja los controles de play, stop, pause, marcas, eventos, etc y está formado por tres componentes que se describen a continuación:

2.5.1.3.1. Debriefing Events: El componente Debriefing Events se encarga de identificar los eventos dentro de la información alojada en el componente de memoria compartida Data Prd. Con esta información, el componente ofrece la posibilidad de llevar a cabo un Debriefing por eventos, es decir, que la reproducción *"salte"* de un evento a otro.

2.5.1.3.2. Rprd Ctrl: El componente Rprd Ctrl se encarga de gestionar las marcas introducidas por el usuario, incrementar y decremento la velocidad de reproducción, gestionar el *"salto"* a un tiempo específico del vuelo, detener, pausar, retroceder y avanzar la reproducción. Además le permite al usuario moverse entre

una marca y otra, hacia adelante o hacia atrás y mantiene el estado actual de la reproducción evitando el envío de múltiples comandos del mismo tipo.

2.5.1.4. SYNC MNG

El componente SYNC MNG se encarga de tomar el tiempo UTC y los datos almacenados en el componente de memoria compartida Data Prd y ordenarlos, con respecto al tiempo, de manera secuencial. Una vez que ha ordenado los datos, el componente se queda a la espera de una acción proveniente del componente de control de reproducción Rprd Ctrl. Finalmente, cuando el componente dispone de los datos ordenados y la acción que se origina en el control de reproducción, procede al envío/detención de datos al paquete de visualización VLZT MNG y queda a la espera de siguientes comandos. Cabe aclarar que la toma de datos de la memoria compartida solo se lleva a cabo una vez por cada Debriefing.

2.5.1.5. SHARED MEM MNG

El paquete SHARED MEM MNG se encarga de mantener toda la información de entrada con la finalidad de ofrecer en forma corporativa a los componentes que acceden de forma concurrente.

2.5.1.5.1. Sorted Local Data: Este componente se encarga de tomar los datos provenientes de la aeronave y almacenarlos en una memoria compartida. Además, se encarga de controlar la escritura en las distintas áreas de memoria utilizando mecanismos de sincronización tales como semáforos, monitores y secciones críticas. Mantiene los datos del área de mensajes del SD para su posterior almacenamiento externo.

2.5.2. Diseño general de los componentes de software

2.5.2.1. Diseño del componente SYNC MNG

Como se aprecia en la figura 2.12.

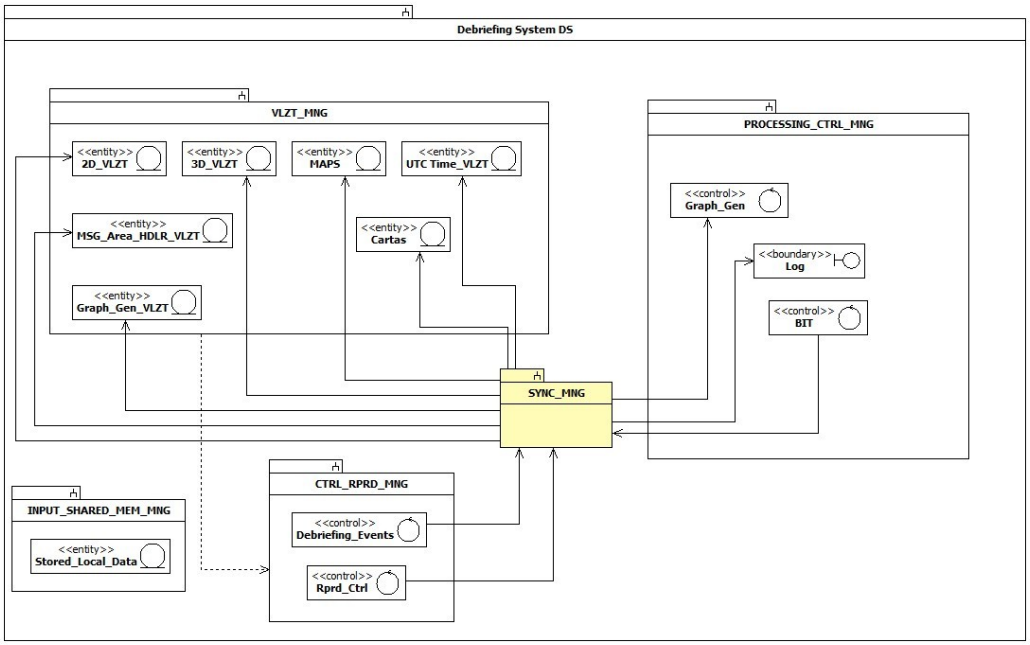

Figura 2.12: Componente SYNC MNG

- Características físicas: El componente SYNC MNG, se representa mediante un paquete de software que contiene un conjunto de funciones encargadas de llevar a cabo la sincronización de los datos de entrada y enviarlos de forma concurrente a los componentes de visualización que corresponda.
- Características lógicas: Se hace uso de los componentes predefinidos por el lenguaje de programación, tales como ordenación, concurrencia, a través de threads, utilización de patrones de diseño como singleton y metodologías de sincronización como semáforos[1].

2.5.2.1.1. Propósito: El propósito del componente es llevar a cabo las tareas de sincronización de tal forma que la reproducción del vuelo se encuentre correctamente sincronizada con respecto al tiempo establecido en los datos de entrada. Además, el componente se encarga de llevar a cabo la generación de gráficos de parámetros de vuelo seleccionables por el usuario respecto al tiempo.

[1]https://es.wikipedia.org/wiki/Semáforo_(informática)

2.5.2.1.2. Función: El componente SYNC MNG básicamente se encarga de llevar a cabo dos tareas: la sincronización de los datos de entrada y la generación de gráficos (cuando corresponde). Para esto, el componente ejecuta las siguientes funciones:

- Posterior al arranque del SD, el componente SYNC MNG lee los datos de la memoria compartida (Input Local Data), ordena estos datos con respecto al tiempo de vuelo (UTC), esto es, los sincroniza para luego almacenar los datos sincronizados nuevamente en la memoria compartida (Common Local Data).
- Finalizada la sincronización, el componente se encarga de inspeccionar en la memoria compartida si se deben generar gráficos.
- Una vez que se llevo a cabo la sincronización y la generación de gráficos (si fuera necesario), el componente se queda a la espera de comandos provenientes del paquete CTRL RPRD MNG. Una vez recibido un comando, el componente procede a la carga de parámetros de configuración (almacenados en la memoria compartida) necesarios para el inicio de la reproducción del vuelo.
- Al recibir el comando play (se describe la inicialización de un vuelo) el componente genera un *pull de threads*. El número de threads está definido por el parámetro Set of VLZT components to destination. Estos threads se encargan de alimentar a los componentes de visualización a una tasa aproximada y variable que ronda los 100 milisegundos generalmente, aunque esta tasa se puede modificar (es configurable desde el Rprd Ctrl).
- Iniciado el pull de threads, el componente SYNC VLZT se queda a la espera de comandos referentes a la reproducción del vuelo (stop, pause, etc) provenientes del componente Rprd Ctrl. En caso de recibir el comando pause, el componente congela la ejecución del pull de trheads mientras que el comando stop la detiene por completo y el componente se queda a la escucha de comandos.
- Si se inició la reproducción del vuelo, el usuario la detiene y desea comenzar con la reproducción del mismo vuelo, el componente no vuelve a sincronizar los datos y a calcular los gráficos seleccionados por el usuario (cuando corresponda).

- Comunicar al componente MSG Area HDLR VLZT todos los mensajes de error referentes al debriefing. Es importante remarcar que el componente MSG Area HDLR VLZT solo administrará mensajes de error referentes al debriefing y solo algunos mensajes de error del SD.
- Comunicar al componente Log todos los errores, eventos y warnings referentes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (log).
- Responder al componente BIT, ante la solicitud, el estado actual del componente Rprd Ctrl.

2.5.2.1.3. Dependencias: Este componente es utilizado solo por el paquete de control de reproducción. Los datos recibidos son procesados y enviados únicamente al paquete de visualización.

2.5.2.1.4. Interfases: A continuación se presenta el diagrama de flujo del componente SYNC MNG especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente posee dos fases de operación, la primera ocurre al iniciar la reproducción del vuelo y la segunda, al finalizar la fase anterior (Figura 2.13).

2.5.2.1.5. Recursos: El componente SYNC MNG hace uso de los siguientes recursos: Paquete Memoria Compartida (SHARED MEM MNG).

2.5.2.2. Diseño del componente Rprd Ctrl

Diseño del Componente (Figura 2.14).

- Características físicas: El componente Rprd Ctrl contiene un conjunto de funciones encargadas de interpretar y ejecutar los comandos introducidos por el usuario a través del componente Rprd Ctrl VLZT, e interpretar y ejecutar los parámetros configurados por el usuario que se encuentran almacenados en la memoria.

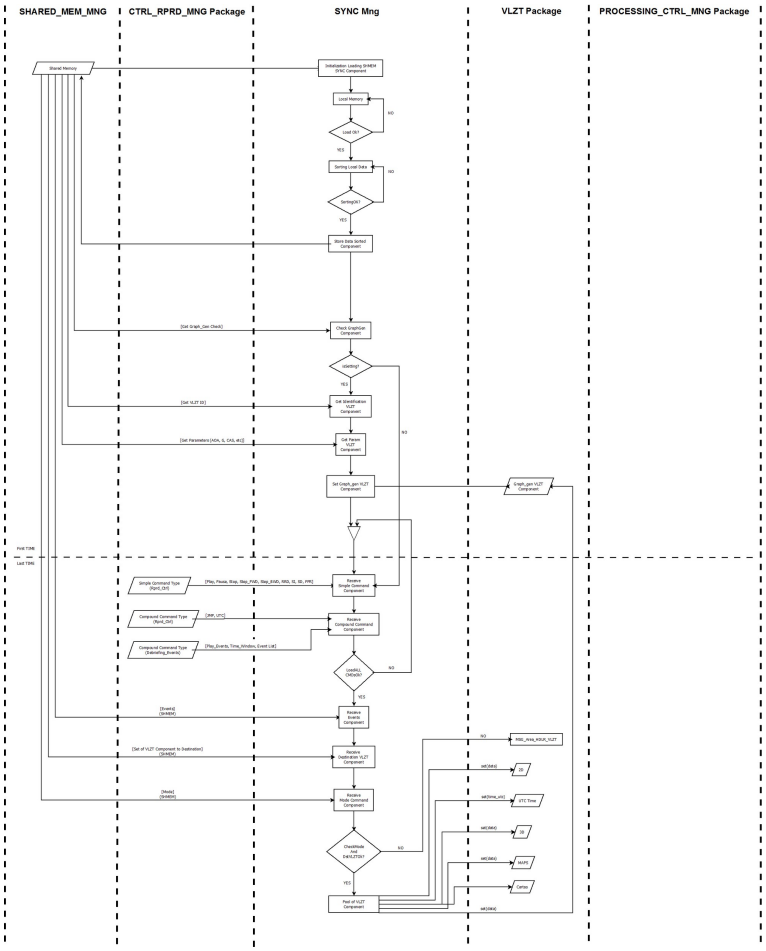

Figura 2.13: Componente SYNC MNG

- Características lógicas: Gestionar la sincronización de los datos con respecto al tiempo y a las acciones (Play, Pause, Stop, etc.) realizadas por el usuario del SD.

2.5.2.2.1. Propósito: El propósito del componente es administrar la reproducción de los vuelos y brindar todas las funciones de control de reproducción permitiendo la interacción entre el usuario y el SD.

2.5.2.2.2. Función: El componente se encarga de gestionar la reproducción de los vuelos interpretando comandos y ejecutando los parámetros de configuración introducidos por el operador del SD. Para esto el componente lleva a cabo las siguientes funciones:

- Interpreta los comandos introducidos por el usuario desde el componente Rprd

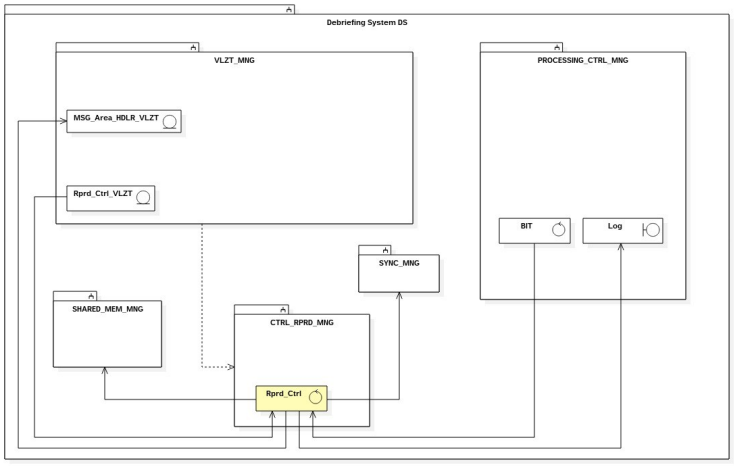

Figura 2.14: Diseño del Componente Rprd Ctrl

Ctrl VLZT para luego entregar este comando al componente SYNC MNG.

- Verificar las configuraciones llevadas a cabo por el usuario, almacenadas en la memoria compartida por el componente particular, la verificación se lleva a cabo sobre los parámetros componente SYNCMNG, ya que este no lleva a cabo comprobación.
- En caso de que el usuario desee introducir una marca de tiempo durante la reproducción del vuelo (Desde el formulario Set Mark), el componente Rprd Ctrl se encarga de introducir dicha marca a la UTC seleccionada por el usuario en la memoria compartida (Common Local Data).
- Enviar al componente SYNC MNG el comando introducido por el usuario ya sea simple o compuesto.
- Enviar un mensaje con los cambios de estado al componente MSG Area HDLR VLZT para informar al usuario sobre dicho cambio (por ejemplo, “Reproduction paused”).
- Comunicar al componente MSG Area HDLR VLZT todos los mensajes de error referentes al debriefing. Es importante remarcar que el componente MSG Area HDLR VLZT solo administrará mensajes de error referentes al debriefing y solo algunos mensajes de error del SD.
- Comunicar al componente Log todos los errores, eventos y warnings referen-

tes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (log).

- Responder al componente BIT, ante la solicitud, el estado actual del componente Rprd Ctrl.

2.5.2.2.3. Dependencias: Este componente se utiliza por el componente Rprd Ctrl VLZT. Los datos recibidos son validados y luego enviados al componente SYNC MNG.

2.5.2.2.4. Interfases: A continuación se presenta el diagrama de flujo del componente Rprd Ctrl especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente siempre se encuentra a la espera de comandos provenientes del componente Rprd Ctrl VLZT. Una vez recibido un comando, el componente verifica la integridad de ciertas configuraciones almacenadas en la memoria compartida y envía el comando recibido al componente SYNC MNG (Figura 2.15).

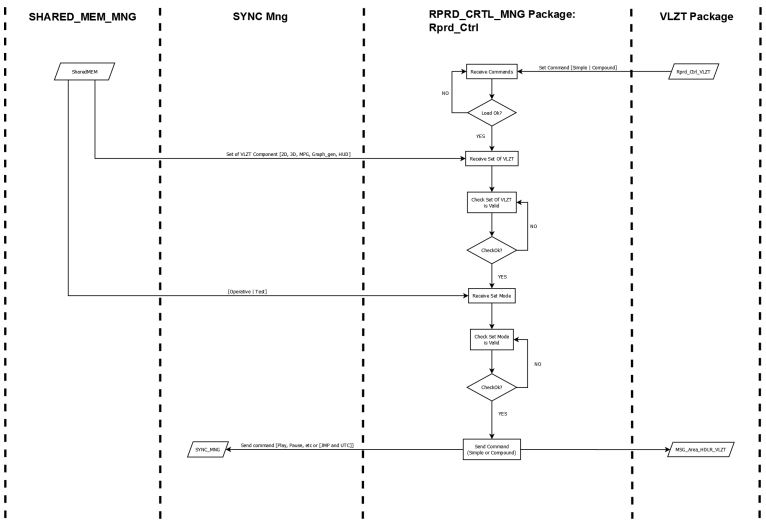

Figura 2.15: Diagrama de flujo del componente Rprd Ctrl

2.5.2.2.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.3. Diseño del componente Debriefing Events

Diseño del componente (Figura 2.16).

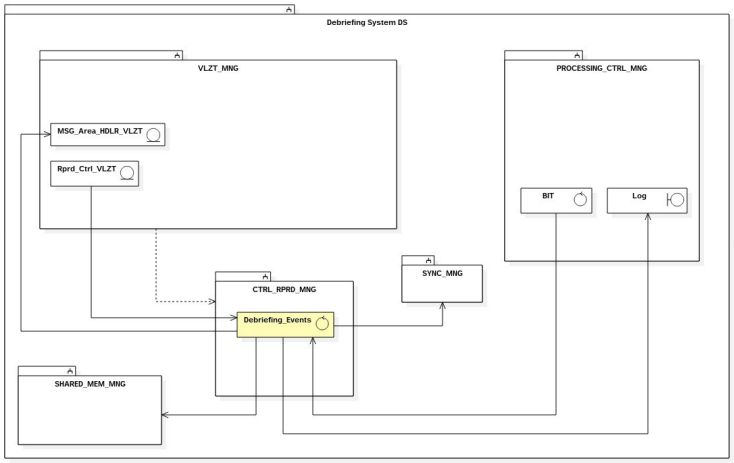

Figura 2.16: Diseño del componente Debriefing Events

- Características físicas: El componente Debriefing Events contiene un conjunto de funciones encargadas de posibilitar la reproducción del vuelo por eventos. Básicamente, el componente se encarga de recopilar la información necesaria para la reproducción del vuelo evento a evento y entregarla al componente SYNC MNG. Es importante tener en cuenta que una vez iniciada la reproducción, el control de la misma es delegado al componente Rprd Ctrl.
- Características lógicas: El componente Debriefing Events contiene un conjunto de funciones, en este caso clases genéricas y primitivas de sincronismo para coordinar el tiempo de reproducción con el tiempo exacto de la ocurrencia de un evento realizando así la tarea de reproducción en dicho punto.

2.5.2.3.1. Propósito: El propósito del componente es brindar la funcionalidad de debriefing por eventos. Esto es, llevar a cabo la reproducción del vuelo pero "saltando" de un evento a otro. Para esto, el componente se encarga de recopilar toda la información necesaria para luego proporcionar esta información al componente SYNC MNG.

2.5.2.3.2. Función: El componente se encarga de brindar la funcionalidad de reproducción del vuelo por eventos interpretando comandos y ejecutando los parámetros de configuración introducidos por el operador del SD. Una vez que se ejecutó la reproducción "evento a evento", el control de la reproducción del vuelo se convierte en responsabilidad del componente Rprd Ctrl.

2.5.2.3.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.3.4. Interfases: A continuación se presenta el diagrama de flujo del componente Debriefing Events especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente siempre se encuentra a la espera del comando (Debrief Events) proveniente del componente Rprd Ctrl VLZT. Una vez recibido este comando, el componente verifica la integridad de ciertas configuraciones almacenadas en la memoria compartida, envía el comando (Play Events) al componente SYNC MNG y delega el control de la reproducción del vuelo al componente Rprd Ctrl (Figura 2.17).

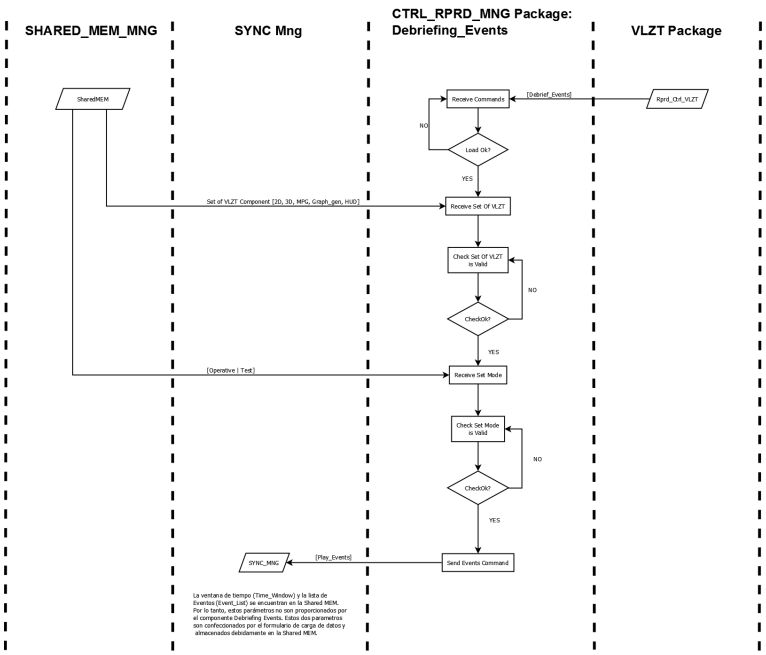

Figura 2.17: Diagrama de flujo del componente Debriefing Events

2.5.2.3.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.4. Diseño del componente Sorted Local Data

Diseño del componente (Figura 2.18).

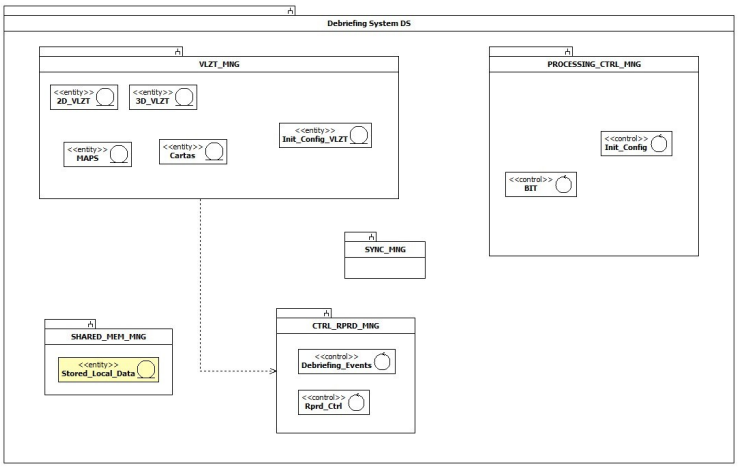

Figura 2.18: Diseño del componente Sorted Local Data

- Características físicas: El componente Sorted Local Data contiene dos tablas de memoria en las que se almacena toda la información gestionada por el SD. Una de estas tablas, solo accesible para lectura, se encarga de almacenar los datos provenientes del GPS, mientras que la otra, accesible para lectura y escritura, se encarga de almacenar los datos necesarios para la configuración de los diversos componentes del SD y los datos que hayan sido procesados y requieran almacenamiento para un uso posterior.
- Características lógicas: Encargado a través de dos listas en memoria dinámica clasificar los elementos según su tiempo de arribo, ordenado en forma ascendente para realizar la reproducción.

2.5.2.4.1. Propósito: El propósito del componente Sorted Local Data consiste en gestionar dos tablas en memoria que contienen todos los datos necesarios para la reproducción del vuelo.

2.5.2.4.2. Función: El componente Sorted Local Data se encarga de crear y gestionar dos tablas de datos en memoria.

2.5.2.4.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.4.4. Interfases: A continuación se presenta el diagrama de flujo del componente Sorted Local Data especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente se encarga de crear y gestionar dos tablas en memoria las cuales se encuentran disponibles en todo momento para el resto de los componentes del SD. Una de estas tablas, de acceso solo para lectura, almacena la información procesada de los datos de de vuelo adquiridos en la aeronave mientras que la otra, de acceso tanto para lectura como para escritura, almacena todo tipo de datos. Esto es, parámetros de configuración o bien los datos de vuelo que requieran algún tipo de procesamiento para luego ser utilizados (Figura 2.19).

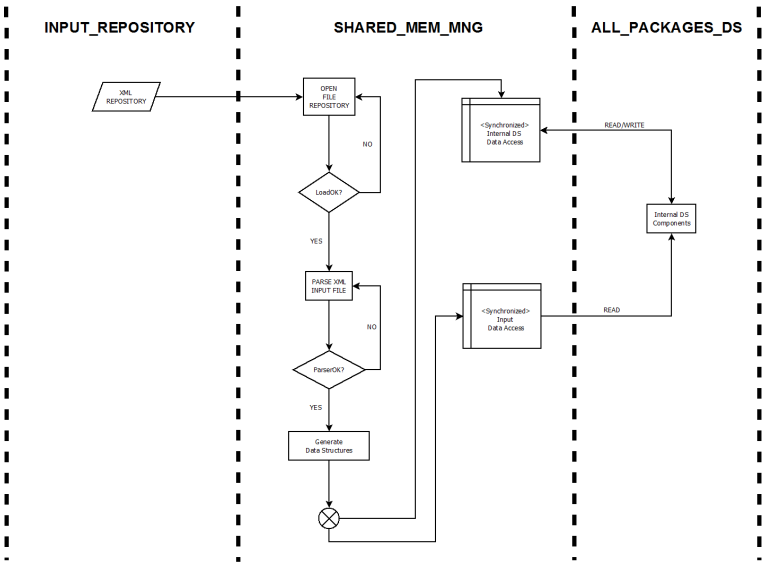

Figura 2.19: Diagrama de flujo del componente Sorted Local Data

2.5.2.4.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.5. Diseño del componente Scht Generator

Diseño del componente (Figura 2.20).

- Características físicas: El componente Scht Generator es un componente que, ante la recepción de un comando, genera y almacena una captura de pantalla

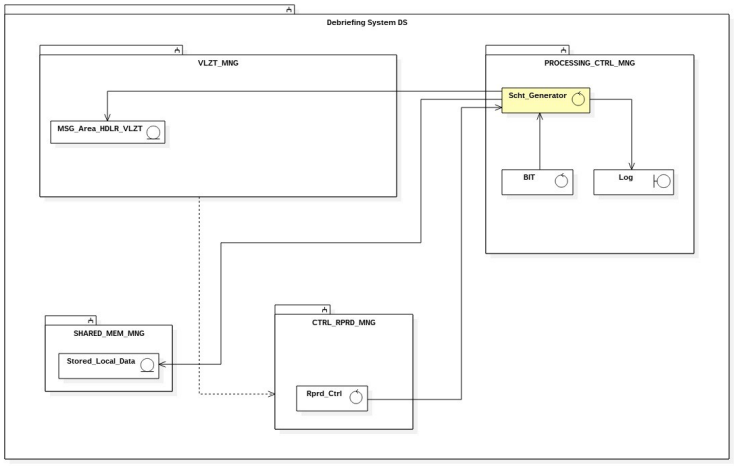

Figura 2.20: Diseño del componente Scht Generator

en el disco duro del host en el que se está ejecutando el SD.

- Características lógicas: A través de primitivas del SD operativo otorgada por clases de objetos, realiza la captura de pantalla sobre el SD y la almacena en la ruta (especificada) correspondiente.

2.5.2.5.1. Propósito: El propósito del componente es proporcionar la funcionalidad de generación de capturas de pantalla.

2.5.2.5.2. Función: El componente Scht Generator se encarga de generar y almacenar una captura de pantalla del SD. Para esto, el componente tiene a cargo las siguientes funcionalidades:

- Recibir e interpretar el comando [Get Screenshot] proveniente de la barra de herramientas del SD.
- Comprobar el estado de la herramienta, esto es, si el usuario pulso el botón de la barra de herramientas para tomar una captura de pantalla y lo vuelve a presionar varias veces de forma reiterada el componente solo generará una captura.
- Obtener la captura de pantalla para finalmente generar el archivo de salida y almacenarlo en el disco duro del host en donde se ejecuta el SD.

- Comunicar al componente MSG Area HDLR VLZT todos los mensajes de error referentes al debriefing. Es importante remarcar que el componente MSG Area HDLR VLZT solo administrará mensajes de error referentes al debriefing y solo algunos mensajes de error del SD.
- Comunicar al componente Log todos los errores, eventos y warnings referentes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (Log).
- Responder al componente BIT, ante la solicitud, el estado actual del componente Rprd Ctrl.

2.5.2.5.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.5.4. Interfases: A continuación se presenta el diagrama de flujo del componente Scht Generator especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente se encarga de generar y almacenar capturas de la pantalla del SD (Figura 2.21).

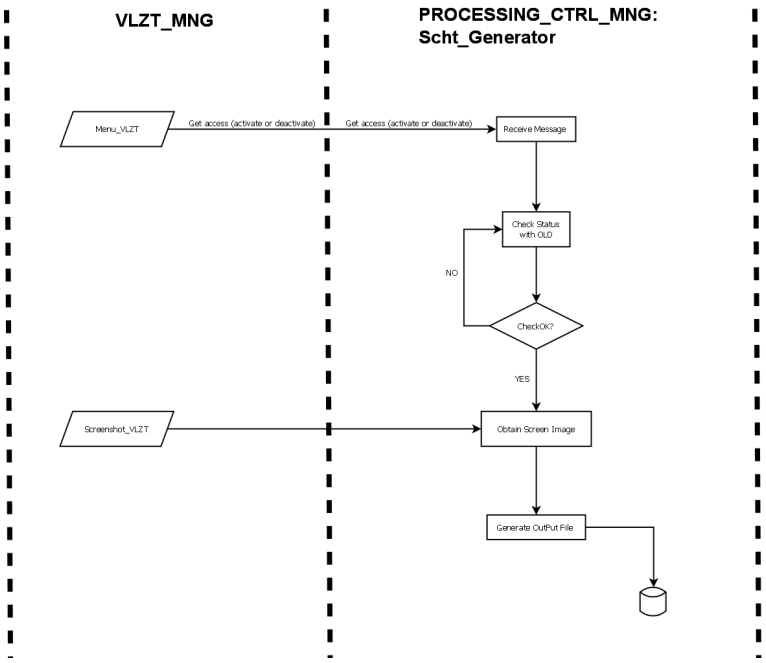

Figura 2.21: Diagrama de flujo del componente Scht Generator

Puede observarse en el diagrama que el componente se ejecuta al recibir un

comando (Get Screenshot) proveniente de la barra de herramientas del SD. Si el estado es correcto, el componente obtiene la captura de pantalla del SD, genera el archivo de salida y finalmente lo almacena en el disco duro del host en donde el SD se encuentra en ejecución. Las capturas de pantalla son almacenadas en el directorio "C:/InstallPath/Screenchots" y el nombre asignado por el componente a cada captura es "Screenshot xxx" donde xxx es un número que parte de 000 a 999.

2.5.2.5.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.6. Diseño del componente Init Config

Diseño del componente (Figura 2.22).

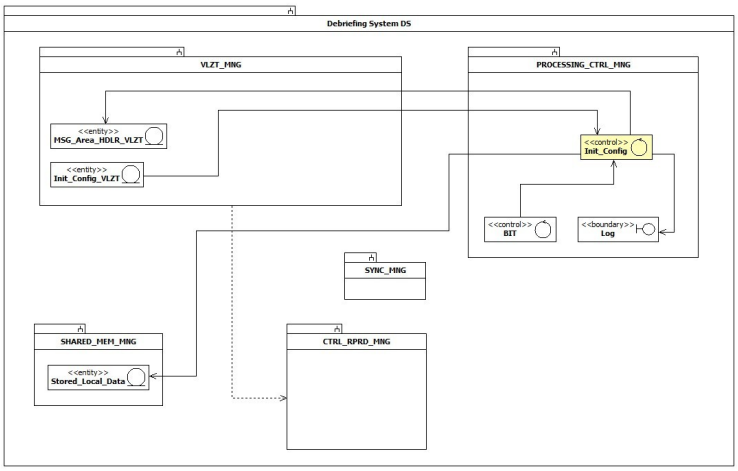

Figura 2.22: Diseño del componente Init Config

- Características físicas: El componente Init Config está representado por un conjunto de funciones encargadas de tomar los parámetros de configuración del SD almacenados en un archivo y tomar los parámetros de configuración introducidos por el usuario desde el componente Init Config VLZT para almacenar estos parámetros en la memoria compartida (tabla Common Local Data).
- Características lógicas: No posee.

2.5.2.6.1. Propósito: El propósito del componente Init Config es gestionar el almacenamiento en la memoria compartida (tabla Common Local Data) de todos los parámetros de configuración del SD.

2.5.2.6.2. Función: El componente Init config se encarga de administrar el almacenamiento de las configuraciones del SD en la memoria compartida. Para esto ejecuta las siguientes funciones:

- Obtener parámetros de configuración del componente Init Config VLZT.
- Verificar la consistencia de los datos obtenidos del componente Init Config VLZT. Si hay un error, el componente intentará obtener los datos en cinco ocasiones y si el error persiste, el componente envía un mensaje de error simple "Error al cargar los datos de configuración" al componente MSG Area Hdlr VLZT para notificar al usuario de este error, un mensaje de error detallado al componente Log y finalmente detiene la ejecución del vuelo.
- Si los datos obtenidos del componente Init Config VLZT son correctos, el componente procede al almacenamiento de dichos datos en la memoria compartida (tabla Common Local Data).
- Obtener parámetros de configuración del SD por defecto.
- Verificar la consistencia de los datos obtenidos del componente. Si hay un error, el componente intentará obtener los datos en cinco ocasiones y si el error persiste, el componente envía un mensaje de error simple al componente MSG Area Hdlr VLZT para notificar al usuario de este error, un mensaje de error detallado al componente Log y finalmente detiene la ejecución del vuelo.
- Si los datos obtenidos son correctos, el componente procede al almacenamiento de dichos datos en la memoria compartida (tabla Common Local Data).
- Comunicar al componente MSG Area HDLR VLZT todos los mensajes de error referentes al debriefing. Es importante remarcar que el componente MSG Area HDLR VLZT solo administrará mensajes de error referentes al debriefing y solo algunos mensajes de error del SD.

- Comunicar al componente Log todos los errores, eventos y warnings referentes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (log).
- Responder al componente BIT, ante la solicitud, el estado actual del componente.

2.5.2.6.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.6.4. Interfases: A continuación se presenta el diagrama de flujo del componente Init Config especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente se encarga de llevar a cabo el almacenamiento de todos los parámetros de configuración necesarios por el SD en la memoria compartida (Figura 2.23).

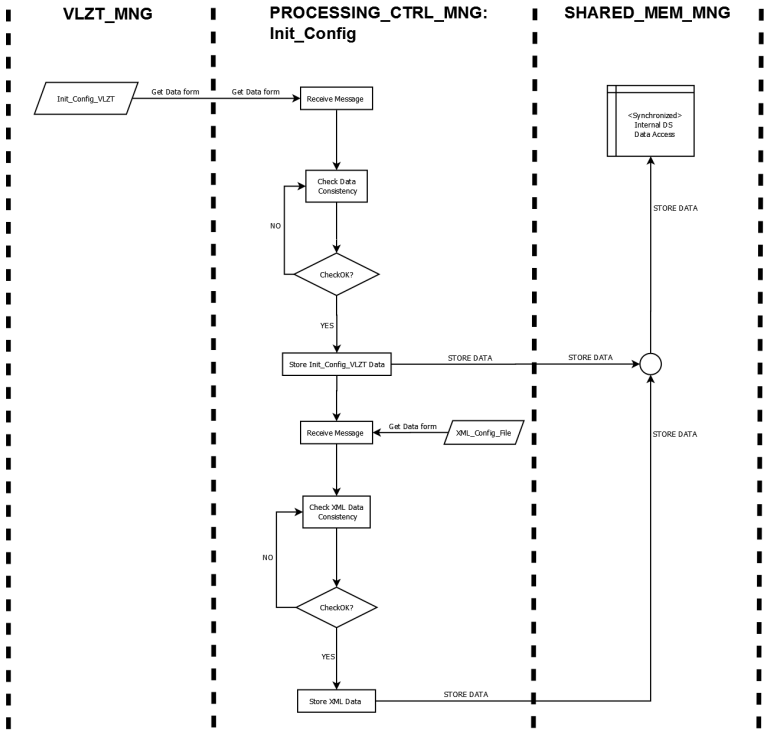

Figura 2.23: Diagrama de flujo del componente Init Config

Si bien en el diagrama es posible observar que la primera acción del componente es obtener los datos de configuración provenientes del componente Init Config VLZT, en realidad esto no es así. La primera acción del componente es obtener los

datos de configuración por defecto del SD y almacenar estos datos en la memoria compartida. Esto se debe a que el componente Init Config VLZT requiere de estos datos para su correcta ejecución. Es importante remarcar que al cargar los datos de configuración, se lleva a cabo una validación de los mismos. Si estos datos no son correctos, el componente reintenta la carga en cinco ocasiones y si el error persiste, se descartan los datos, se envía un mensaje de error simple "Error al cargar los datos de configuración" al componente MSG Area Hdlr VLZT, un mensaje de error detallado al componente Log y se detiene la reproducción del vuelo. Si la carga de los datos es correcta, entonces el componente procede a su almacenamiento en la memoria compartida (tabla Common Local Data). Una vez que se cargaron los parámetros de configuración del SD (por defecto), el componente procede a la obtención de los datos proporcionados por el componente Init Config VLZT para luego llevar a cabo una verificación de la consistencia de los mismos. Si estos datos no son correctos, el componente reintenta la carga en cinco ocasiones y si el error persiste, se descartan los datos, se envía un mensaje de error simple "Error al cargar los datos de configuración" al componente MSG Area Hdlr VLZT, un mensaje de error detallado al componente Log y se detiene la reproducción del vuelo. Si la carga de los datos es correcta, entonces el componente procede a su almacenamiento en la memoria compartida (tabla Common Local Data).

2.5.2.6.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.7. Diseño del componente BIT

Diseño del componente (Figura 2.24).

- Características físicas: El componente BIT está representado por un conjunto de funciones encargadas de determinar el SOH de todos los componentes del SD para garantizar su correcto funcionamiento.
- Características lógicas: Realizar la lista de verificación de todos los componentes para el correcto funcionamiento y reportando las anomalías, a través, de pasajes de mensajes entre threads.

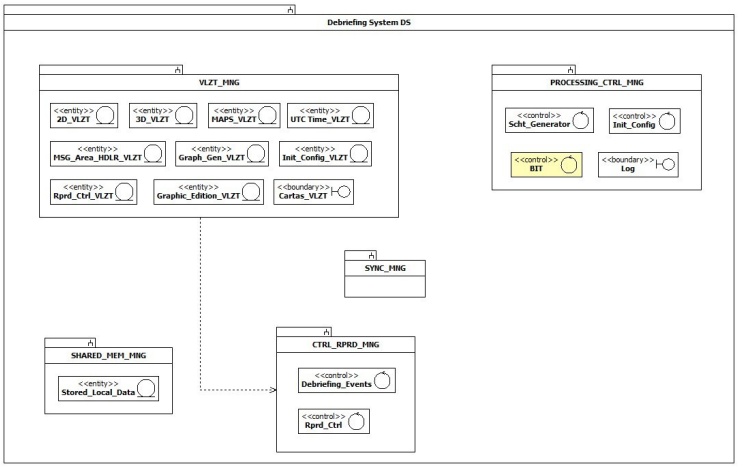

Figura 2.24: Diseño del componente BIT

2.5.2.7.1. Propósito: El propósito del componente BIT es llevar a cabo una prueba de funcionamiento de los módulos de software al iniciarse el SD. Determina si cada uno de los componentes se encuentra en condiciones de ejecutarse correctamente y, por conjunción, determinar el estado del SD completo (FULL OP o DEGRADED). También registra el estado del SD (detallando los módulos con error) en el componente Log.

2.5.2.7.2. Función: El componente debe determinar el SOH del SD. Para esto, el componente ejecuta las siguientes funciones:

- Solicitar el estado SOH a cada uno de los componentes del SD.
- Recibir y evaluar el SOH de cada componente.
- Si el total de los SOH correcto, el componente genera un reporte (System FULL OP) y lo envía al componente Log. Si algún SOH es incorrecto, el componente vuelve a solicitar el estado a todos aquellos módulos cuyo SOH no es satisfactorio en cinco oportunidades. Si algún SOH continúa siendo incorrecto, el componente genera un reporte (System DEGRADATED), lo envía al componente Log, notifica al usuario que hay un error al iniciar el SD "Error al iniciar el SD" mediante una ventana emergente y detiene la ejecución del mismo.

- Comunicar al componente Log todos los errores, eventos y warnings referentes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (Log).

2.5.2.7.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.7.4. Interfases: A continuación se presenta el diagrama de flujo del componente BIT especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente se encarga de solicitar el *"State of Healt"* de cada uno de los componentes del SD y, a partir de la conjunción de estos, generar un reporte que finalmente es enviado al componente Log (Figura 2.25).

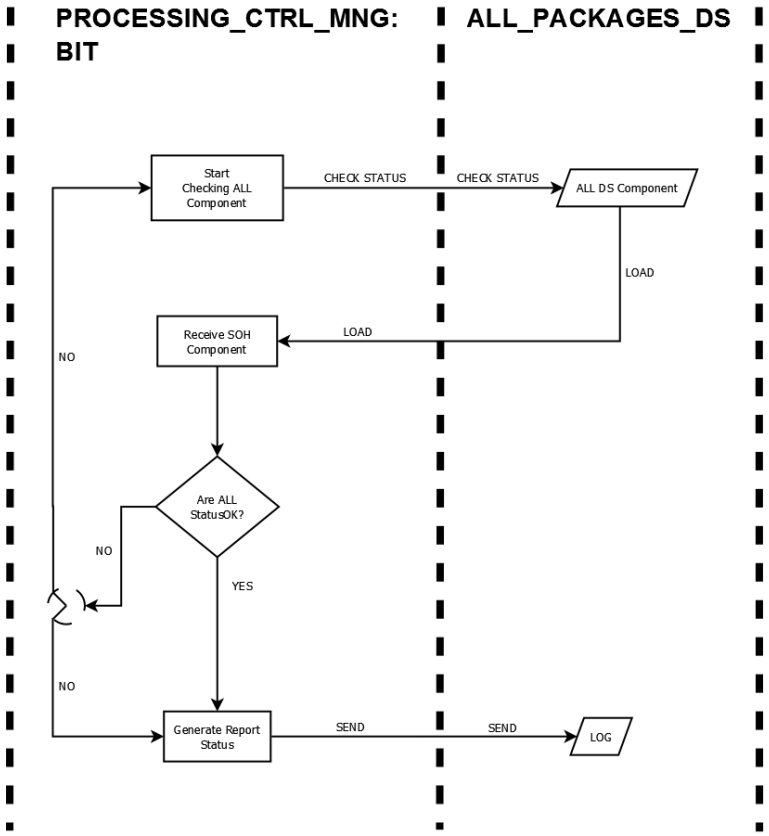

Figura 2.25: Diagrama de flujo del componente BIT

Como puede observarse en el diagrama, Al iniciarse el SD el componente comienza a solicitar el SOH al resto de los componentes. El recibir todos los SOH solicitados, el componente evalúa la conjunción de los mismos. Si uno o más de los

SOH es incorrecto, el componente vuelve a solicitar el estado a los componentes cuyos SOH resultaron insatisfactorios en cinco oportunidades. Si al menos uno de los SOH es insatisfactorio, el componente genera un reporte (System DEGRADATED) detallando el o los componentes que presentaron un SOH incorrecto, lo envía al componente Log, notifica al usuario del SD mediante una ventana emergente con el mensaje "Error al iniciar el SD" y detiene la ejecución del mismo. Si la conjunción de los SOH es correcta, es decir, todos los SOH son correctos, el componente genera un reporte (System FULL OP) y lo envía al componente Log.

2.5.2.7.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.8. Diseño del componente Log

Diseño del componente (Figura 2.26).

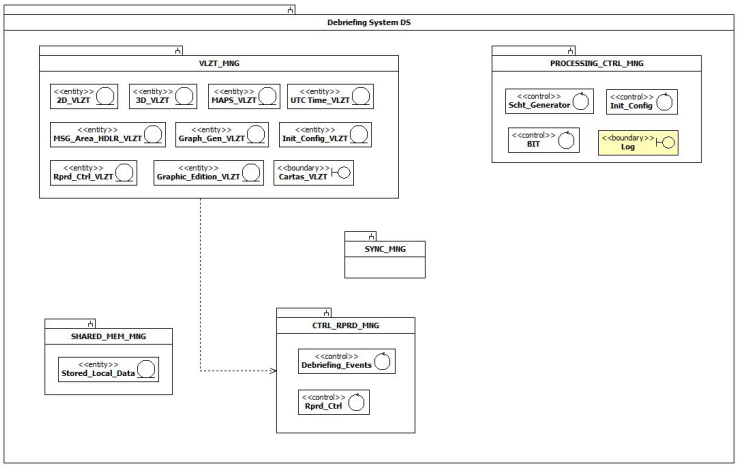

Figura 2.26: Diseño del componente Log

- Características físicas: El componente Log está representado por un conjunto de funciones encargadas de atender constantemente las solicitudes generadas por el resto de los componentes del SD registrar sus eventos.
- Características lógicas: No posee.

2.5.2.8.1. Propósito: El propósito de este componente es llevar un registro de eventos del SD en un archivo. Este archivo puede ser utilizado por los encargados

del mantenimiento del SD para controlar el funcionamiento general del mismo, para diagnóstico y resolución de anomalías.

2.5.2.8.2. Función: El componente Log se encarga de la gestión de la totalidad de los mensajes de error, warning o caution generados por el resto de los componentes de software que conforman el SD. Para esto, el componente ejecuta las siguientes funciones:

- Generar el archivo Log del SD.
- Recibir los mensajes provenientes de los distintos componentes.
- Comprobar la sintaxis de los mensajes recibidos.
- Si la sintaxis de los mensajes es correcta abrir el archivo log y almacenar dicho mensaje. Si la sintaxis es incorrecta, descartar el mensaje.
- Responder al componente BIT, ante la solicitud, el estado actual del componente SOH.

2.5.2.8.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.8.4. Interfases: A continuación se presenta el diagrama de flujo del componente Log especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente gestiona los mensajes de error provenientes de cualquiera de los componentes que integran el SD para finalmente almacenar estos mensajes en un archivo de registro (Log) (Figura 2.27).

Se puede observar que el componente siempre se encuentra a la espera de un mensaje proveniente de uno de los restantes componentes del SD. Al recibir un mensaje de error, el componente comprueba su sintaxis para verificar la integridad del mismo. La sintaxis de un mensaje de error o Log Message es la siguiente:

<Message [ToD] [Type] [Level] [Error Code]> Text </Message>

- ToD: Time of Day.
- Type: Tipo de mensaje. Existen tres tipos, a cada uno de los cuales se les asigna un color dependiendo de su importancia (Green, Yellow, Red).

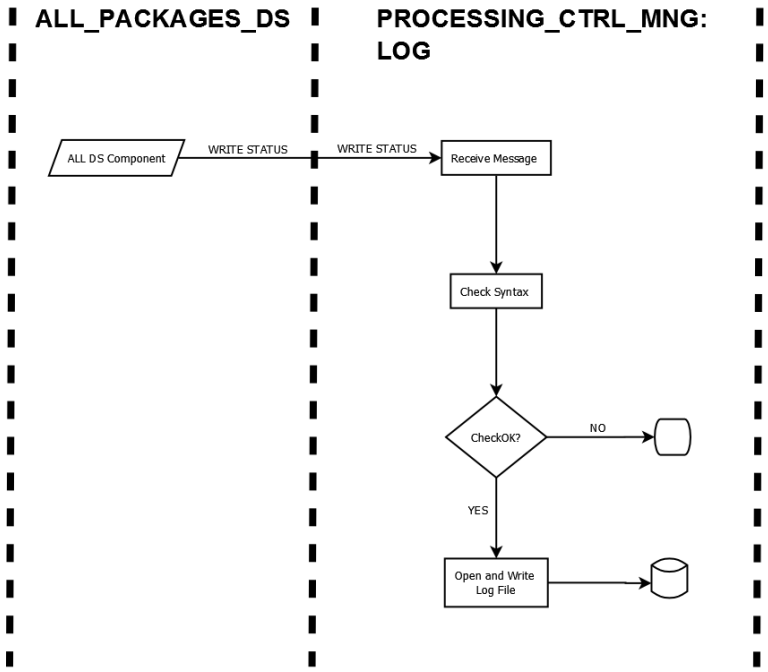

Figura 2.27: Diagrama de flujo del componente Log

- Level: Nivel de ocurrencia del error.
- ErrorCode: Codigo de error. Este atributo es opcional.
- Text: Mensaje de error enviado por los distintos componentes del SD. Los componentes, de forma opcional, pueden adjuntar al mensaje de error el tiempo de ocurrencia del mismo. El formato del cuerpo del mensaje de error es el siguiente: [Event Time] Text

Si el componente de procedencia del mensaje de error no introdujo el tiempo de ocurrencia del mismo, el componente Log introduce el valor de ToD en el atributo Event Time. Si la sintaxis del mensaje es incorrecta, el componente descarta el mensaje y se queda a la espera de otro mensaje de error. Si la sintaxis del mensaje es correcta, el componente abre el archivo de registro (Log) introduce el mensaje de error, lo cierra y se queda a la espera del siguiente mensaje de error.

2.5.2.8.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.9. Diseño del componente 2D VLZT

Diseño del componente (Figura 2.28).

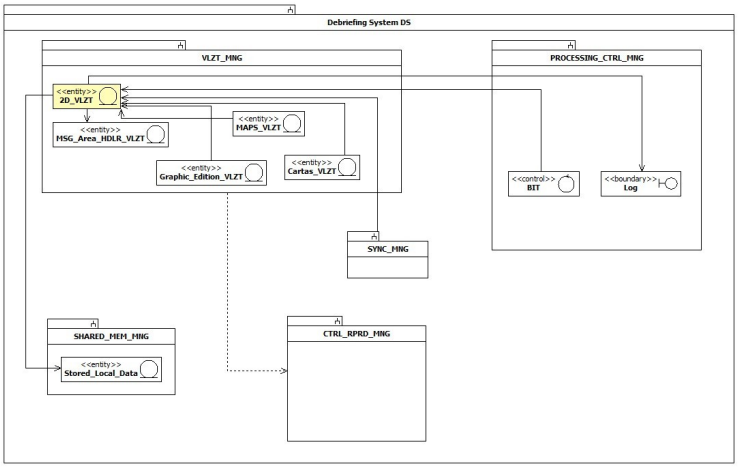

Figura 2.28: Diseño del componente 2D VLZT

- Características físicas: El componente 2D VLZT está representado por un conjunto de funciones encargadas de gestionar la representación del vuelo en dos dimensiones.
- Características lógicas: No posee.

2.5.2.9.1. Propósito: El propósito del componente es recibir parámetros de vuelo (Lat, Long, etc.) para luego representar dichos parámetros utilizando un icono correspondiente a la aeronave en cuestión sobre un mapa en dos dimensiones.

2.5.2.9.2. Función: No aplica a este componente.

2.5.2.9.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.9.4. Interfases: Corresponde a la figura 2.29.

2.5.2.9.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.10. Diseño del componente 3D VLZT

Diseño del componente (Figura 2.30).

- Características físicas: El componente 3D VLZT está representado por un conjunto de funciones encargadas de gestionar la representación del vuelo en

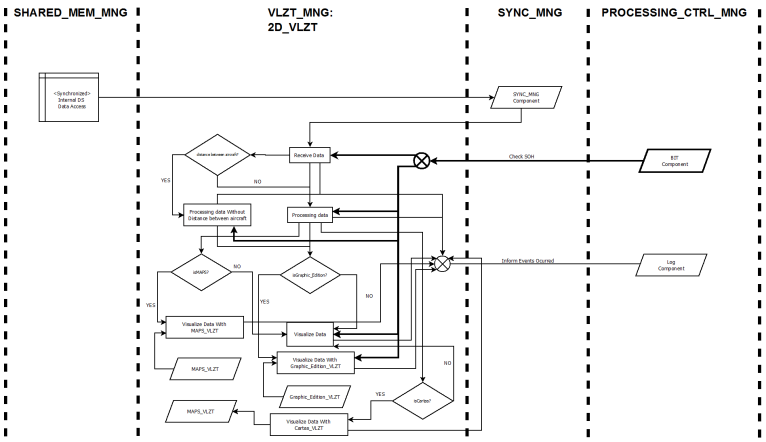

Figura 2.29: Diagrama de flujo del componente 2D VLZT

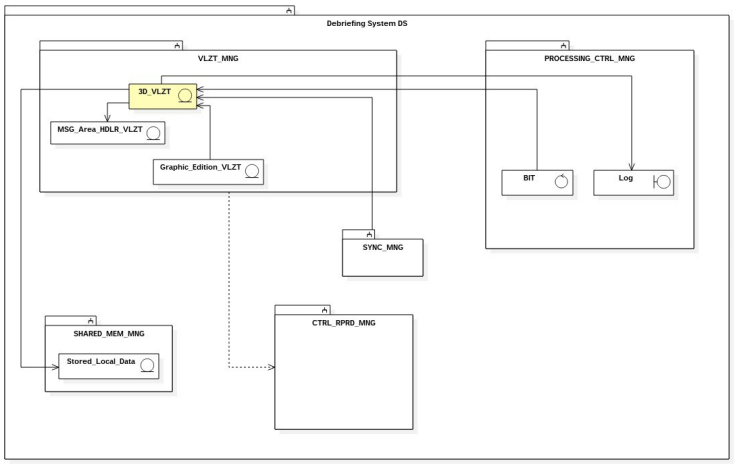

Figura 2.30: Diseño del componente 3D VLZT

tres dimensiones.

- Características lógicas: No posee.

2.5.2.10.1. Propósito: El propósito del componente es recibir parámetros de vuelo (Lat, Long, elev, etc.) para luego representar dichos parámetros utilizando un icono correspondiente a la aeronave en cuestión dentro de un terreno en tres dimensiones, los archivos DEM para el calculo de elevación del terreno, como así los distintos modelos de SARM de aeronaves como ser A4AR, IA-63 Pampa, IA-58 Pucará, Lama, etc.

2.5.2.10.2. Función: No aplica a este componente.

2.5.2.10.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.10.4. Interfases: Corresponde a la figura 2.31.

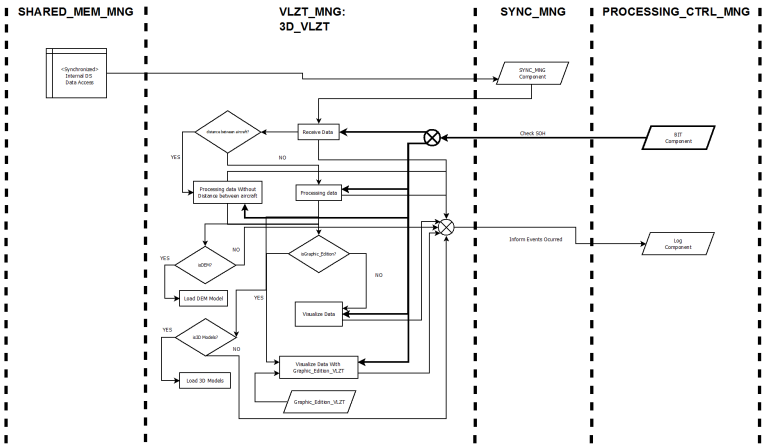

Figura 2.31: Diagrama de flujo del componente 3D VLZT

2.5.2.10.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.11. Diseño del componente UTC Time VLZT

Diseño del componente (Figura 2.32).

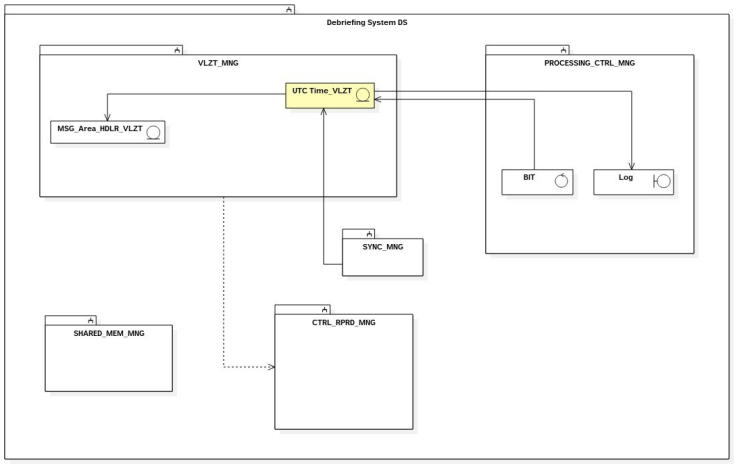

Figura 2.32: Diseño del componente UTC Time VLZT

- Características físicas: El componente UTC Time VLZT está representado por un conjunto de funciones encargadas de mostrar por pantalla la hora de vuelo.

- Características lógicas: No posee.

2.5.2.11.1. Propósito: El propósito del componente es la visualización de la hora de vuelo en formato HH:MM:SS.

2.5.2.11.2. Función: El componente UTC Time VLZT básicamente se encarga de la visualización del tiempo de vuelo. Para esto, debe llevar a cabo las siguientes funciones:

- Recibir el tiempo del vuelo (UTC) del componente SYNC MNG.
- Convertir este tiempo en formato HH:MM:SS para finalmente mostrarlo por pantalla.
- Comunicar al componente Log todos los errores, eventos y warnings referentes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (Log).
- Responder al componente BIT, ante la solicitud, el estado actual del componente SOH.

2.5.2.11.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.11.4. Interfases: A continuación se presenta el diagrama de flujo del componente UTC VLZT especificando la transformación de los datos de entrada y como son entregados a la salida. El funcionamiento del componente es simple, solo se encarga de recibir el tiempo de vuelo UTC y mostrarlo en formato HH:MM:SS (Figura 2.33).

Se puede observar en el diagrama que el componente posee dos entradas de datos y una salida. Una de las corresponde a la hora de vuelo en formato UTC, enviada por el componente SYNC MNG. Al recibir este dato, el componente se encarga de convertirlo a formato HH:MM:SS (Processing data) para finalmente mostrarlo por pantalla (Visualize Data). Además, al recibir un dato UTC, el componente verifica la integridad del mismo y en caso de error (por ejemplo, "UTC Time VLZT: Error en el formato UTC recibido") lo registra en el componente Log. Otra entrada

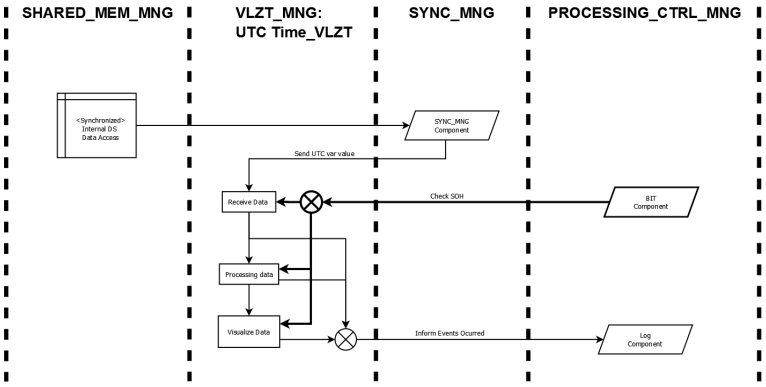

Figura 2.33: Diagrama de flujo del componente UTC Time VLZT

proviene del componente BIT, en donde este último solicita el estado de la recepción de datos, procesamiento de datos y visualización de datos del componente MPG VLZT. Finalmente, la única salida del componente corresponde a los errores que el componente UTC Time VLZT envía al componente Log.

2.5.2.11.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.12. Diseño del componente MSG Area HDLR VLZT

Diseño del componente (Figura 2.34).

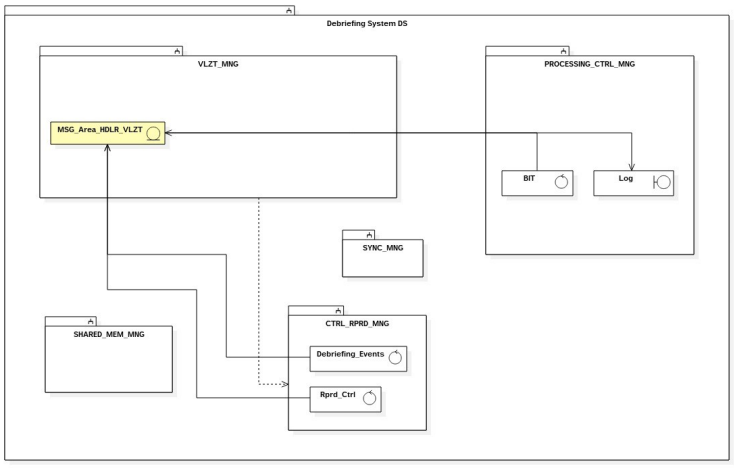

Figura 2.34: Diseño del componente MSG Area HDLR VLZT

- Características físicas: El componente MSG Area HDLR VLZT está representado por un conjunto de funciones encargadas de recibir y mostrar por pantalla

los mensajes de error provenientes del resto de los componentes de software del SD.

- Características lógicas: No posee.

2.5.2.12.1. Propósito: El propósito del componente es gestionar los mensajes de error provenientes de los distintos módulos del SD para informar al usuario de los mismos en forma simple y ordenada.

2.5.2.12.2. Función: El componente MSG Area HDLR VLZT se encarga de la gestión de los mensajes de error que serán mostrados por pantalla. Para esto, debe llevar a cabo las siguientes funciones:

- Recibir los mensajes de error provenientes de los distintos componentes del SD.
- Agregar la hora del SD a cada mensaje.
- Colorear los mensajes de error en color rojo.
- Comunicar al componente Log todos los errores, eventos y warnings referentes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (Log).
- Responder al componente BIT, ante la solicitud, el estado actual del componente SOH.

2.5.2.12.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.12.4. Interfases: A continuación se presenta el diagrama de flujo del componente MSG Area HDLR VLZT especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente se encarga de recibir los mensajes de error provenientes de los distintos componentes que integran el SD, agregar la hora del SD a los mismos y colorear (rojo) los mensajes de error (Figura 2.35).

Se observa en la figura, el componente posee dos entradas de datos y una salida. Una de las entradas corresponde a los mensajes de estado (no tienen por qué ser

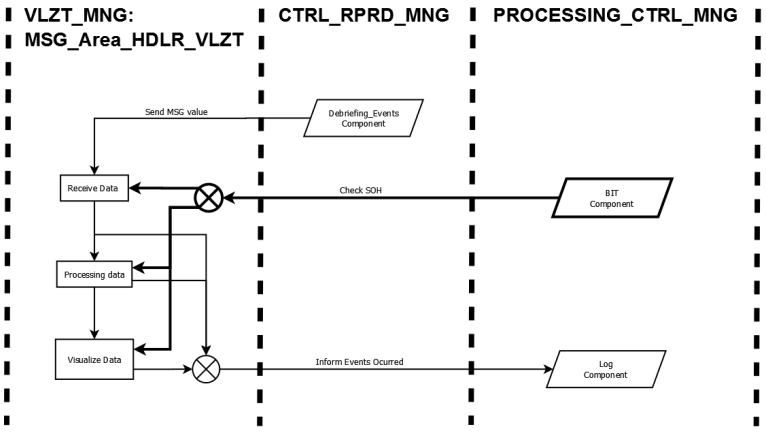

Figura 2.35: Diagrama de flujo del componente MSG Area HDLR VLZT

de error) provenientes de cualquier componente del SD. Al recibir este dato, el componente se encarga agregar la hora del SD al mensaje y colorearlo en caso de que sea un mensaje de error (Processing data) para finalmente mostrarlo por pantalla (Visualize Data). Además, al recibir los mensajes, el componente verifica la integridad de los mismos y en caso de error, lo registra en el componente Log. Otra entrada proviene del componente BIT, en donde este último solicita el estado de la recepción de datos, procesamiento de datos. Finalmente, la única salida del componente corresponde a los errores que el componente UTC Time VLZT envía al componente Log.

2.5.2.12.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.13. Diseño del componente Graph Gen VLZT

Gestor de gráficos de parámetros de vuelo versus el tiempo (Figura 2.36).

- Características físicas: No posee.
- Características lógicas: No posee.

2.5.2.13.1. Propósito: No aplica a este componente.

2.5.2.13.2. Función: No aplica a este componente.

2.5.2.13.3. Dependencias: No posee dependencias de otros componentes.

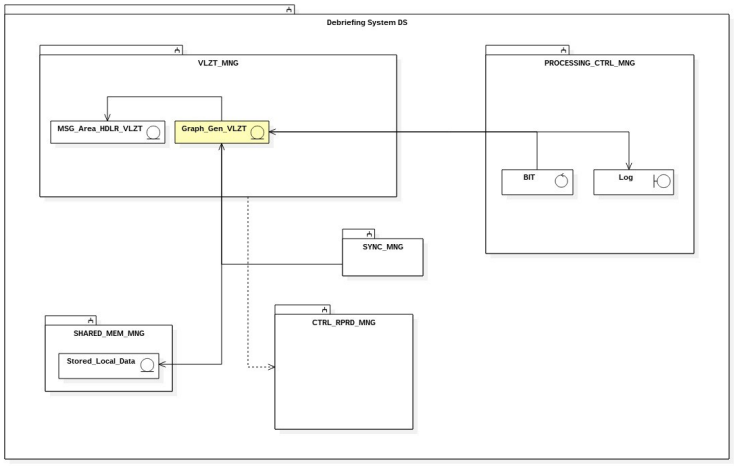

Figura 2.36: Diseño del componente Graph Gen VLZT

2.5.2.13.4. Interfases: Se presenta la interfaz de este componente en la 2.37.

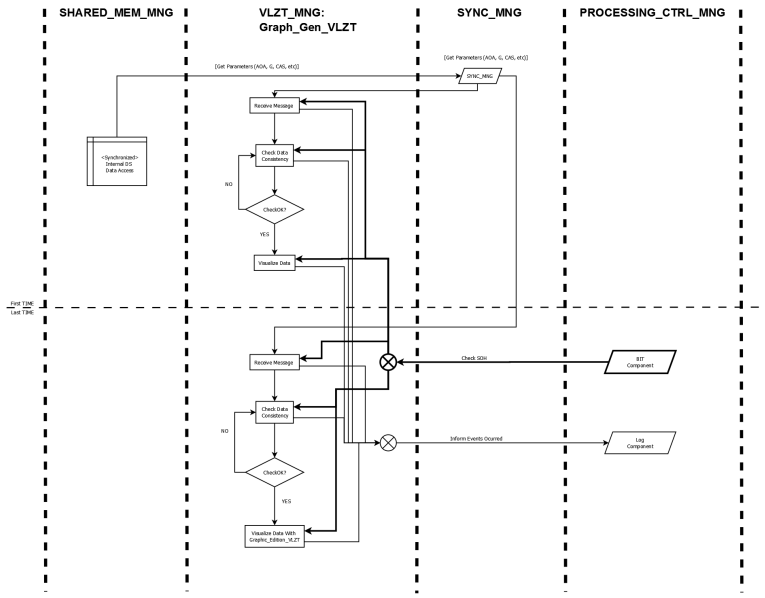

Figura 2.37: Diagrama de flujo del componente Graph Gen VLZT

2.5.2.13.5. ZipUtil: Esta clase tiene como responsabilidad la compresión y descompresión de archivos utilizando el formato de compresión ZIP. Para esto, la clase ofrece los siguientes métodos:

- Comprimir: cuando los argumentos de entrada del método son una ruta a un directorio, un filtro, el nombre de archivo zip y un parámetro de ejecución

(crearAuto), la tarea del método es comprimir los archivos del directorio indicado y almacenarlos en un archivo zip. El filtro se encarga de indicar los ficheros del directorio que se deben seleccionar. Si directorio es una cadena (string) vacía, el filtro indicará el archivo a comprimir (sólo ese). Si crearAuto = True, zipfile será el directorio en el que se almacenara el archivo zip, el cual se generará automáticamente el nombre con la fecha y hora actual.

- Comprimir: cuando los argumentos de entrada del método son un array de cadenas (string) que contiene los archivos a comprimir, el nombre del archivo zip resultante y un parámetro de ejecución (crearAuto), el método se encarga de comprimir los archivos del arreglo en el archivo zip indicado. Si crearAuto = True, zipfile será el directorio en el que se almacenará el archivo zip, el cual se generará automáticamente con la fecha y hora actual como nombre.
- Descomprimir: este método se encarga de descomprimir el contenido del archivo zip especificado como parámetro de entrada (zipFic) en el directorio indicado (directorio). Si zipFic no tiene la extensión .zip, se entenderá que es un directorio y se procesará el primer fichero .zip de ese directorio. Si el parámetro de entrada eliminar es True se eliminará ese fichero zip después de descomprimirlo y si el parámetro renombrar es True se añadirá al final la extensión .descomprimido.
- Contenido: la tarea de este método es devolver el contenido del archivo zip indicado como parámetro de entrada (zipFic).

2.5.2.13.6. Recursos: No utiliza recursos de otros componentes.

2.5.2.14. Diseño del componente Rprd Ctrl VLZT

Diseño del componente (Figura 2.38).

- Características físicas: El componente Rprd Ctrl VLZT está representado por un conjunto de funciones encargadas de comunicar al componente Rprd Ctrl las acciones que el usuario desea ejecutar sobre la reproducción del vuelo.
- Características lógicas: No posee.

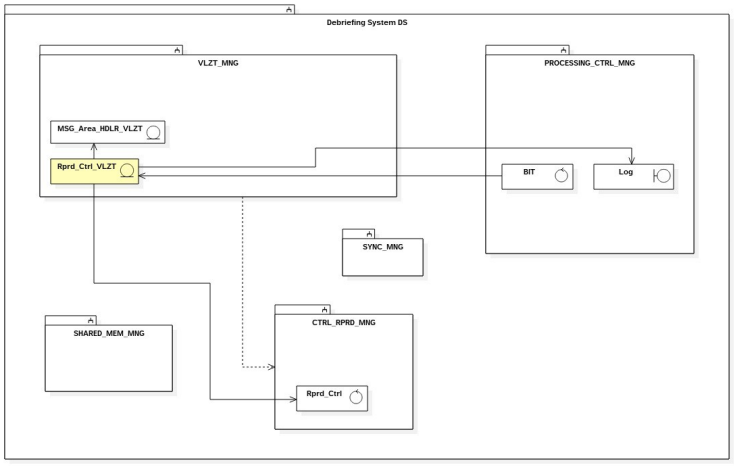

Figura 2.38: Diseño del componente Rprd Ctrl VLZT

2.5.2.14.1. Propósito: El propósito del componente es brindar una interfaz gráfica entre el componente Rprd Ctrl encargado del control de la reproducción del vuelo y el usuario.

2.5.2.14.2. Función: El componente Rprd Ctrl VLZT se encarga de comunicar al componente Rprd Ctrl las acciones que el usuario desea ejecutar sobre la reproducción del vuelo. Para esto, debe llevar a cabo las siguientes funciones:

- Capturar el evento generado por el botón oprimido por el usuario.
- Analizar el evento en cuestión.
- Registrar la acción (evento) que el usuario desea ejecutar sobre la reproducción del vuelo.
- Comunicar dicho evento al componente Rprd Ctrl.
- Comunicar al componente Log todos los errores, eventos y warnings referentes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (Log).
- Responder al componente BIT, ante la solicitud, el estado actual del componente SOH.

2.5.2.14.3. Dependencias: No posee dependencias de otros componentes.

2.5.2.14.4. Interfases: A continuación se presenta el diagrama de flujo del componente Rprd Ctrl VLZT especificando la transformación de los datos de entrada y como son entregados a la salida. Básicamente el componente se encarga de brindar una interfase entre el control de la reproducción del vuelo y el usuario permitiendo que este sea capaz de controlar dicha reproducción (Figura 2.39).

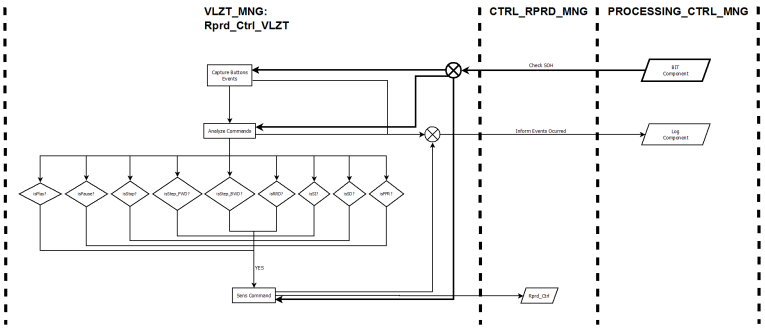

Figura 2.39: Diagrama de flujo del componente Rprd Ctrl VLZT

Como se observa en el diagrama, el funcionamiento del componente es simple. El componente posee dos entradas y dos salidas. Una de las entradas corresponde a los eventos generados por los botones del panel de control de reproducción del SD (play, pause, stop, etc.). Según el botón presionado por el usuario, el componente se encarga de generar el comando correspondiente y enviarlo al componente Rprd Ctrl encargado del control de la reproducción del vuelo, constituyéndose de esta manera, una de las salidas del componente. La segunda entrada corresponde al componente BIT el cual se encarga de solicitar el estado SOH del componente para determinar el estado global del SD. Finalmente, la segunda salida está formada por los eventos registrados en el componente Log del SD. Estos eventos no solo se traducen en errores o warnings sino que también incluyen los cambios de estado (play a pause, etc.).

2.5.2.14.5. Recursos: No utiliza recursos de otros componentes.

2.5.2.15. Diseño del componente Graphic Edition VLZT

Diseño del componente (Figura 2.40).

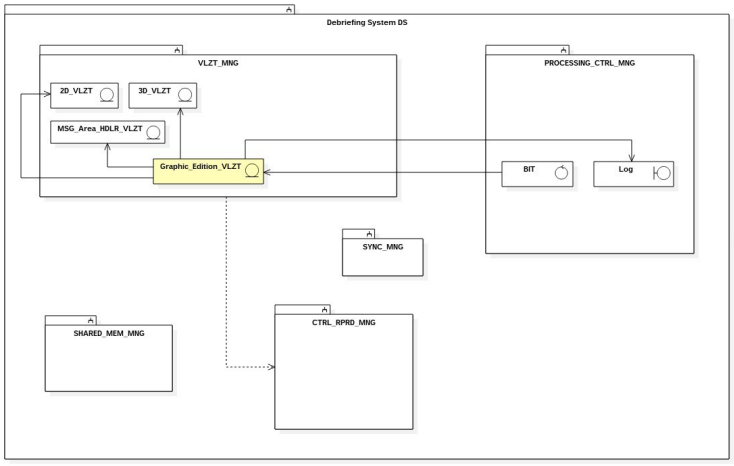

Figura 2.40: Diseño del componente Graphic Edition VLZT

- Características físicas: El componente Graphic Edition VLZT está representado por un conjunto de funciones encargadas de permitir que el usuario realice gráficas sobre los componentes 2D VLZT y 3D VLZT utilizando el puntero del mouse.
- Características lógicas: No posee.

2.5.2.15.1. Propósito: El propósito del componente es brindar al usuario la posibilidad de llevar a cabo gráficas mediante el puntero del mouse, sobre la representación del vuelo en 2D y 3D.

2.5.2.15.2. Función: El componente Graphic Edition VLZT se encarga de permitir que el usuario pueda llevar a cabo gráficas utilizando el puntero del mouse sobre los componentes 2D VLZT y 3D VLZT. Para esto, debe llevar a cabo las siguientes funciones:

- Comunicar al componente Log todos los errores, eventos y warnings referentes al funcionamiento interno del componente. Luego, el componente Log se encarga de volcar estos datos en el archivo de registro de eventos (Log).
- Responder al componente BIT, ante la solicitud, el estado actual del componente SOH.

2.5.2.15.3. Interfases: A continuación se presenta el diagrama de flujo del componente Graphic Edition VLZT especificando la transformación de los datos de entrada y como son entregados a la salida (Figura 2.41).

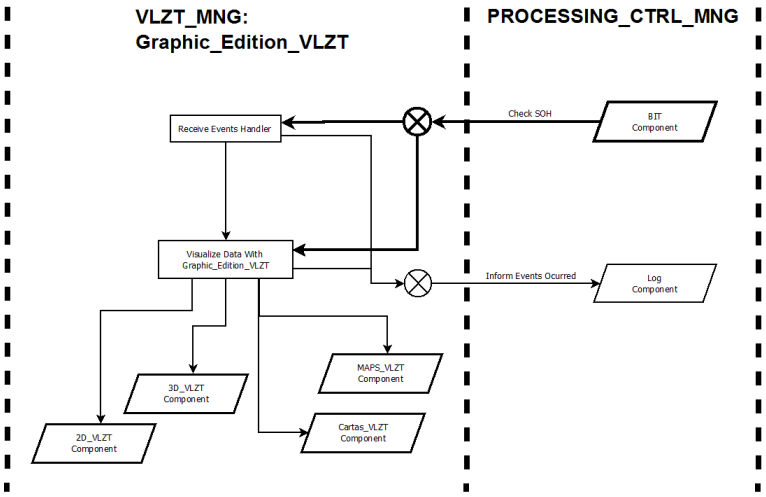

Figura 2.41: Diagrama de flujo del componente Graphic Edition VLZT

2.6. Implementación

En este apartado solo se mostrará la implementación basada en el lenguaje UML de los componentes principales y más importantes del SD y se dejarán el resto sin mostrar. Todos los componentes de clases de diseño que se mostraron en la sección anterior, se desarrollaron con la misma metodología. La razón de minimizar la documentación de la presente sección implementación es puramente y exclusivamente por cuestión de espacio, dado que se tornaría demasiado extensivo el presente documento. El componente que se mostrará es el componente SYNC MNG (descripto en la sección 2.5.2.1) en donde se explicó los artefactos de clases de diseño. En esta sección se detallará el diagrama de clases de implementación (que poseen correspondencia directa a las de diseño) que cooperativamente ofrecen las funciones para cumplir con la tarea de sincronización de datos y envío a los correspondientes componentes de visualización (Figura 2.42).

A continuación se muestran ciertas partes del diseño de la figura 2.42 con el único fin de mostrar más en detalle el mismo (Figura 2.43, 2.44 y 2.45).

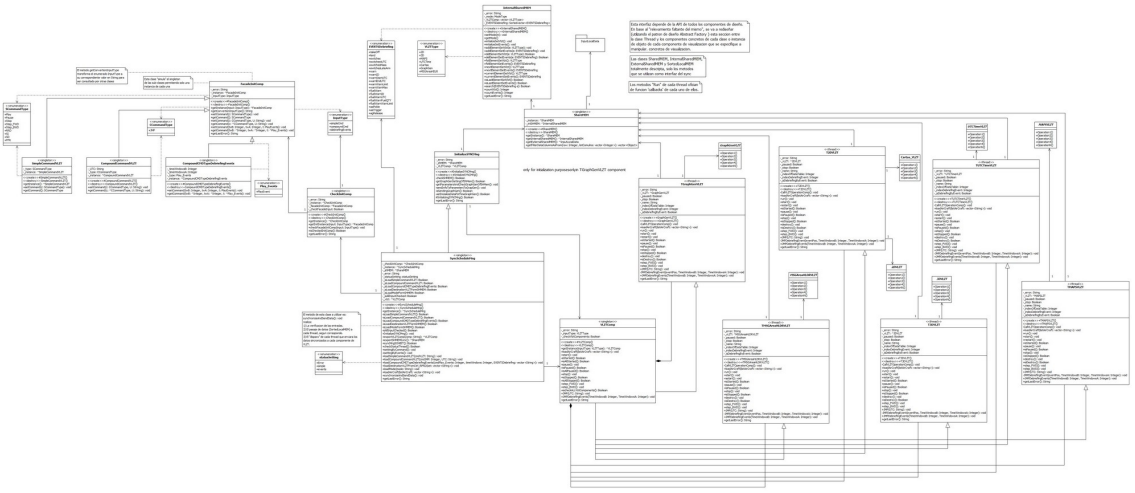

Figura 2.42: Diagrama interno del componente SYNC MNG

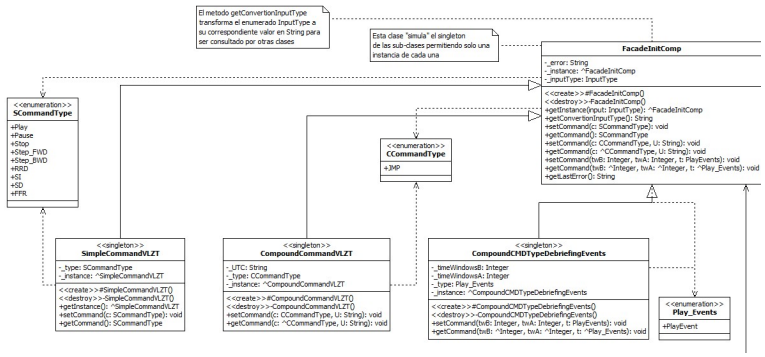

Figura 2.43: Diagrama interno de clases Auxiliares del componente SYNC MNG

Este componente, se compone de las siguientes clases de las cuales se describirán las características de las principales como por ejemplo, relaciones entre ellas, datos internos, responsabilidades, etc. El resto de clases que componen el componente SYNC MNG no se detallaran en esta sección por la razón de que su finalidad es de exponer funcionalidades auxiliares, es decir, no funcionan como entidades independientes sino más bien como subordinadas a las clases principales.

2.6.1. SyncScheduleMng

Esta es la clase principal del componente y define la interfaz de acceso para la carga de datos, la clasificación (sincronización con respecto al tiempo de eventos) de estos y el posterior envío a cada uno de los componentes de visualización. Los métodos que ofrece son:

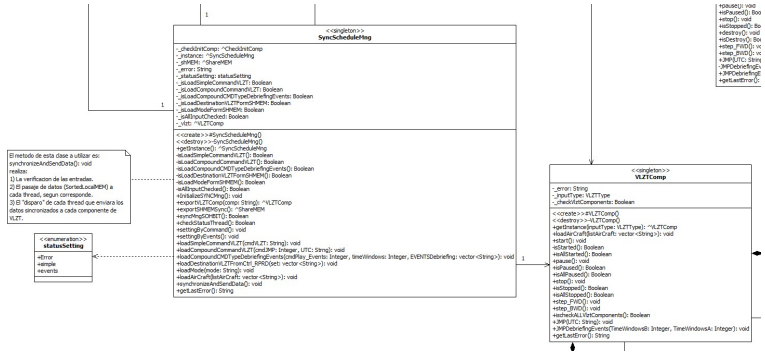

Figura 2.44: Diagrama interno de clases Auxiliares - cont.

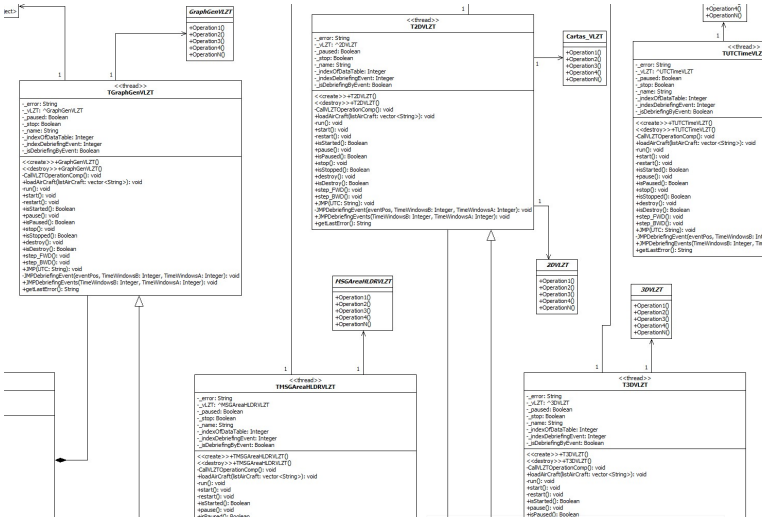

Figura 2.45: Diagrama interno de clases Auxiliares - cont.

- loadMode: Este método permite configurar el modo y toma como parámetro una variable tipo string que representa el modo (O—E).
- loadCommandVLZT: Este método permite configurar el comando elegido y toma como parámetro una variable tipo string que representa a dicho modo.
- loadDestVLZT: Este método permite configurar el o los componentes de destino a enviar datos y toma como parámetro una variable de tipo string que representa el conjunto de estos componentes (separados por coma), por ejemplo: “2D,3D,”.
- synchronizeAndSendData: Este método toma los datos de entrada, e invoca a métodos de la clase SortLocalData para clasificar los datos de entrada y de allí se invoquen a los métodos de la clase CheckInitComp que se encargara de

obtener los datos de la clase ShareMEM y verifica su consistencia. Estos datos son pasados (por parámetro) a la clase SortLocalData quien los clasifica y los deja disponible en la clase SortedLocalMEM que corresponde a un almacén de datos ordenados. Finalmente estos datos son extraídos por este método y en base a la información de destino se envía por medio de la clase VLZTComp a los destinatarios del paquete de visualización.

2.6.2. ShareMEM

Esta clase contiene los datos de entrada, organizados en un almacén de datos, (ya preprocesados) del programa y los deja a disponibilidad a los componentes que así lo requieran. La funcionalidad no es explicada aquí, para más información con respecto a la misma, dirigirse a sección 2.5.2.4.

2.6.3. FacadeInitComp

Esta clase se encarga de "agrupar" a las distintas clases concretas (Mode, CommandVLZT y DestVLZT) las funciones sobre la entrada de datos permitiendo un único acceso sobre estos a través de una interfaz única. Esto puede realizarse gracias a la implementación del patrón de diseño Facade. Los métodos que ofrece son:

- lgetInstance: Este método es el encargado de entregar a la clase CheckInitComp un objeto de alguna de su subclases, dependiendo del parámetro de entrada, esto es, si el parámetro es Mode, se entregara un objeto de la clase Mode con sus correspondientes métodos y así para el resto de subclases. La información que hace referencia a la subclase elegida es almacenada internamente.
- getConvertionInputType: Este método es el encargado de retribuir a la clase CheckInitComp una variable de tipo string que representa la subclase actualmente elegida.

2.6.4. CommandVLZT

La clase en cuestión, representa los comandos a ser enviados a los componentes de visualización que representan las acciones sobre el control de reproducción, por ejemplo, Play, Pause, Stop, etc. Provee para el almacenamiento de los comandos elegidos al método setCommand y para la consulta de cual es actualmente el comando seleccionado, al método getCommand. Sigue el patrón de diseño singleton que permite una y solo una instancia de la clase, creada "on-demand".

2.6.5. DestVLZT

Esta clase almacena el conjunto de componentes de visualización destinos a los cuales se les enviara los datos a visualizar. Para el almacenamiento de los diferentes destinos, se opto por utilizar un vector que contenga a los mismos. Esta clase sigue el patrón de diseño singleton que permite una y solo una instancia de la clase, creada "on-demand". Los métodos que ofrece son:

- initializeSet: Este método permite inicializar y limpiar el vector.
- addElementSet: Este método permite agregar un nuevo elemento al vector, recibiendo como parámetro el elemento en cuestión.
- delElementSet: Este método permite eliminar un elemento existente del vector que es recibido por parámetro. Retorna un valor booleano que representa si la acción fue realizada con éxito o no.
- fstElementSet: Este método es de acceso y permite posicionarse en el primer elemento del vector.
- nextElementSet: Este método es de acceso y permite posicionarse en el siguiente elemento del vector tomando como referencia al actual.
- currentElementSet: Este método es de acceso y permite obtener el elemento actual que indica el índice del vector.
- isLastElementSet: Este método es de acceso y permite consultar si el elemento corriente se corresponde al último elemento del vector.

2.6.6. CheckInitComp

Esta clase es la encargada de obtener las entradas y los datos de la memoria compartida permitiendo verificar la consistencia de los mismo y dejarlos disponibles para la clase SortLocalData quien se encargara de la clasificación. Esta clase sigue el patrón de diseño singleton que permite una y solo una instancia de la clase, creada "on-demand". Los métodos que ofrece son:

- GetInitIntance: Este método, en base al parámetro que recibe, permite obtener un objeto que se corresponde a una de las subclases de FacadeInitComp sobre los datos de entrada (Mode, CommandVLZT o DestVLZT).
- CheckInitComp: Este método permite verificar si todos los datos de entrada recibidos por el objeto de FacadeInitComp son correctos y consistentes.
- CheckSHMEM: Este método permite verificar si los datos almacenados por la clase ShareMEM son correctos y consistentes.
- isCheckInitComp: Este método es de acceso y permite consultar si la verificación realizada por el método CheckInitComp ha sido satisfactoria o no.
- isCheckSHMEM: Este método es de acceso y permite consultar si la verificación realizada por el método CheckSHMEM ha sido satisfactoria o no.
- getShareMEM: Este método es de acceso y permite obtener un objeto de la clase ShareMEM.
- setShareMEM: Este método es de acceso y permite almacenar localmente en la clase CheckInitComp un objeto valido de la clase ShareMEM.

2.6.7. SortLocalData

Esta clase es la responsable de ordenar (sincronizar) teniendo como punto de referencia al tiempo de arribo los datos obtenidos de la clase CheckInitComp sobre información de entrada del SD (shareMEM). La clasificación de datos se realiza una única vez, y son almacenados en un objeto de la clase SortedLocalMEM. Esta clase sigue el patrón de diseño singleton que permite una y solo una instancia de la clase, creada *"on-demand"*. Los métodos que ofrece son:

- Sort: Este método permite ordenar los datos provenientes de la clase ShareMEM en base al tiempo (UTC) que es proporcionado como dato de entrada. Posteriormente a la clasificación, estos datos son enviados al almacén SortedLocalMEM. Esta operación se realiza una sola vez por debriefing.
- isSorted: Este es un método de acceso que permite consultar si los datos fueron correctamente ordenados o ha ocurrido algún error en dicho proceso.
- getInitComp: Este es un método de acceso que permite retribuir un objeto de la clase CheckInitComp solicitada por la clase SyncScheduleMng.
- setInitComp: Este es un método de acceso que permite almacenar localmente en esta clase un objeto de la clase CheckInitComp, con la finalidad de obtener los datos de entrada ya verificados y consistentes.
- getSortedLocalData: Este es un método de acceso que permite retribuir un objeto de la clase SortedLocalMEM para acceder a los datos de entrada ya clasificados con el objetivo de su redistribución a los componentes de visualización.

2.6.8. SortedLocalMEM

Esta clase representa al almacén de datos (local) de la clase ShareMEM con la particularidad que dichos datos han sido clasificados por la clase SortLocalData. Esto datos son utilizados posteriormente por la clase SyncScheduleMng para el posterior envío a los correspondiente componentes de visualización. La clasificación de datos se realiza una única vez, de esta manera los datos son utilizados para los subsiguientes accesos a la misma.

2.6.9. VLZTComp

Esta clase representa el contenedor (*"pool"*) de subprocesos (threads) los cuales son utilizados para en el envío de datos hacia los componentes de visualización destinos. Esta clase es accedida un únicamente por la clase SyncScheduleMng la cual va requiriendo elementos de su contenedor. Los métodos que ofrece son:

- adquire: Este es un método de acceso que permite adquirir un objeto (thread) del contenedor de objetos.
- release: Este es un método de acceso que permite devolver un objeto (thread) del contenedor de objetos.
- isAdquire: Este es un método de acceso que permite verificar si la adquisición ha sido realizada con éxito o ha ocurrido un error.
- isRelease: Este es un método de acceso que permite verificar si la devolución ha sido realizada con éxito o ha ocurrido un error.

Este es el componente más importante del SD dado que maneja toda la funciones principales con respecto a las reproducción de uno o varios vuelos.

2.7. Testing

2.7.1. Introducción

La presente sección del documento trata sobre el análisis de los casos de pruebas realizados al SD. sobre el desempeño con respecto a diferentes ensayos propuestos. Estos ensayos, están orientados a verificar robustez, medidas de casos de fallas, condiciones de fallas no previstas, efectos colaterales, etc.

2.7.2. Alcance

Esta sección del documento, tiene como objetivo de garantizar la calidad de la versión final del SD y asegurar el correcto funcionamiento del mismo en base al desarrollo (esto es las etapas de análisis y diseño de SW) realizado.

2.7.3. Procedimiento

La metodología a utilizar, sera definir correctamente casos de prueba, generar tablas con los resultados de la ejecución de dichos casos de prueba y en base a los resultados obtenidos, una apreciación sobre la satisfacción (o no) entre dichos resultados y el caso de pruebas estipulado.

2.7.3.1. Casos de Uso para Test

Los casos de Uso de Test generalmente comprenden:

- Recopilación de todos los Casos de Uso ya generados.
- Recopilación de documentación generada por el área de desarrollo y la provista por el usuario final.
- Relevar con el usuario final para conocer el funcionamiento de la aplicación, incidente y/o del módulo a probar.

Como resultado se obtiene:

- Casos de Uso de Prueba.
- Planilla de Casos de Prueba.

2.7.3.2. Caso de Test

Por cada Caso de Uso de Prueba puede desprenderse (o no) uno o más casos de prueba.

2.7.3.2.1. Casos de Prueba: Pruebas ejecutadas de uno o más módulos interconectados cuyo resultado permite evaluar si el desarrollo cumple o no con los objetivos. Cada Caso de Prueba deberá indicar la salida o diferentes salidas esperadas.

2.7.3.3. Ejecución del Testing

La Ejecución del Testing, permite: El testing se planifica en base a la relevancia de las modificaciones o del desarrollo que se esté ejecutando, la cantidad de Casos de Prueba y el tiempo para ejecutar los mismos.

2.7.4. TEST

Como se menciono anteriormente, se definirán los siguientes casos de prueba (o test). Para ello, se utilizara como base la siguiente planilla como la que se muestra a continuación:

Nombre del caso de Test	Nombre Completo del casos de Test.
Descripción Abreviada	Descripción breve del caso de test.
ID del caso de Test	Numero que identifica unívocamente al caso de test.
Descripción del caso de Test	Descripción bien completa y concisa.
Condiciones Previas	Campo Opcional.
Relaciones con otros casos de Test	Campo Opcional – Dependencias.
Pasos y condiciones de ejecución	Detallar tipo lista los pasos de ejecución del caso de test.
Resultados esperados	Resultado ideal del SW en relación a los pasos ejecutados.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado • Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW en relación a los pasos ejecutados.
Errores encontrados	Campo Opcional, si los hay identificar y detallar los errores obtenidos.
Responsable de Diseño	Nombre del creador del casos de Test.
Responsable de Ejecución	Nombre del ejecutor del casos de Test creado.
Comentarios	Campo Opcional.

Tabla 2.2: Plantilla de Casos de Test (ejemplo)

2.7.4.1. Ensayos

Cabe como aclaración que los test mostrados, no están ordenados, solo se procede a mostrar los más importantes y relevantes del SD, dejando de lado los menos relevantes, como anteriormente se dijo por cuestiones de espacio dado que son numerosos.

Nombre del caso de Test	Test 1
Descripción Abreviada	Archivos de SUM, Generador de vuelos y Generador de Gráficos.
ID del caso de Test	1,0.
Descripción del caso de Test	Se debe verificar el correcto funcionamiento del SD cuando algunos de estos archivos no están en el directorio establecido por el sistema.
Condiciones Previas	n/a.
Relaciones con otros casos de Test	n/a.
Pasos y condiciones de ejecución	Se ejecuta el SD, con el faltante de uno o varios archivos (SUM, Generador y Graficador).
Resultados esperados	El SD debe reportar tal faltante con un mensaje emergente.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado
	• Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW fueron los esperados, luego de incorporar ventanas emergentes a los faltantes especificados.
Errores encontrados	n/a
Responsable de Diseño	Juan Pablo Rumie Vittar.
Responsable de Ejecución	Juan Oviedo.
Comentarios	s/c.

Tabla 2.3: Plantilla de Casos de Test 1

Nombre del caso de Test	Test 2
Descripción Abreviada	Vuelos Incorrectos.
ID del caso de Test	2,0.
Descripción del caso de Test	Se debe verificar el correcto funcionamiento del SD cuando se carga algún vuelo o ejercicio cuya sintaxis es incorrecta o errónea.
Condiciones Previas	n/a.
Relaciones con otros casos de Test	n/a.
Pasos y condiciones de ejecución	Se ejecuta el SD, con la carga de un archivo sintácticamente incorrecto o erróneo.
Resultados esperados	El SD debe reportar el archivo de vuelo erróneo con un mensaje emergente.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado • Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW en relación al test marco los errores pertinentes en el archivo de entrada especificando la linea en donde se encuentra el error.
Errores encontrados	n/a.
Responsable de Diseño	Juan Oviedo.

Tabla 2.4: Plantilla de Casos de Test 2

Nombre del caso de Test	Test 3
Descripción Abreviada	Carga de 20 Vuelos.
ID del caso de Test	3,0.
Descripción del caso de Test	Se debe verificar la correcta carga de 20 vuelos correctos al SD. El primer vuelo que aparece como “AC1”, además debe tener todos sus parámetros de vuelos seleccionados en la visualización 2D.
Condiciones Previas	n/a.
Relaciones con otros casos de Test	Test 4, debe estar con estado Exitoso.
Pasos y condiciones de ejecución	Se ejecuta el SD, con la carga de un archivo que contiene 20 vuelos en simultaneo.
Resultados esperados	El SD debe realizar la visualización en 2D y en 3D, como así también la mensajería interna y demás componentes, mostrando la información de las 20 aeronaves.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado • Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW en relación a la carga de 20 vuelos que se realizo y visualizo correctamente.
Errores encontrados	n/a
Responsable de Diseño	Ariel Principi.
Responsable de Ejecución	Juan Pablo Rumie Vittar.
Comentarios	s/c.

Tabla 2.5: Plantilla de Casos de Test 3

2.7.4.2. Resumen de los Ensayos

En vista de los casos generales de test y no presentando complicaciones en cuanto a errores del SD, se considera a la fecha como una *versión estable*. Cabe mencionar que futuras actualizaciones del SW, requerirá definir nuevos nuevos requerimientos, casos de uso, artefactos de análisis, diseño e implementación y posteriormente los correspondientes casos de test con su respectiva ejecución como así también (mandatoriamente) la ejecución de los ya vigentes (regresión), generando una nueva

Nombre del caso de Test	Test 4
Descripción Abreviada	Carga de 1 Vuelo.
ID del caso de Test	4,0.
Descripción del caso de Test	Se debe verificar la correcta carga de un (1) vuelo correcto al SD. El primer vuelo que aparece como “AC1”, debe tener todos sus parámetros de vuelos seleccionados en la visualización 2D.
Condiciones Previas	n/a.
Relaciones con otros casos de Test	n/a.
Pasos y condiciones de ejecución	Se ejecuta el SD, con la carga de un archivo que contiene un (1) vuelo únicamente.
Resultados esperados	El SD debe realizar la visualización en 2D y en 3D, como así también la mensajería interna y demás componentes, mostrando la información del único vuelo.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado • Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW en relación a la carga de un vuelo y se visualiza correctamente la leyenda “AC1” en el entorno del SD.
Errores encontrados	n/a.
Responsable de Diseño	Juan Pablo Rumie Vittar.
Responsable de Ejecución	Juan Oviedo.
Comentarios	s/c.

Tabla 2.6: Plantilla de Casos de Test 4

iteración en el ciclo de vida del SD.

Nombre del caso de Test	Test 6
Descripción Abreviada	Agregar/Eliminar Cartografía.
ID del caso de Test	6,0.
Descripción del caso de Test	Se debe verificar que se realiza correctamente la carga y borrado de cartas de navegación desde el componente visual 2D.
Condiciones Previas	n/a.
Relaciones con otros casos de Test	n/a.
Pasos y condiciones de ejecución	Se ejecuta el SD, con la carga de un archivo que contiene varios vuelos o un vuelo, es indistinto y luego se comprueba la correcta carga y borrado sobre el componente visual 2D las cartas de navegación.
Resultados esperados	El SD debe realizar la visualización en 2D de las cartas que se seleccionaron para ser cargadas y poder quitarlas del mismo también, que en dicho caso, solo se visualice de fondo el mapa elegido.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado • Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW en relación a el agregado y posterior eliminación de la cartografía se realiza y visualiza correctamente en el componente visual 2D, incluso el "solapamiento" de varias cartografías, se realiza sin inconvenientes.
Errores encontrados	n/a.
Responsable de Diseño	Ariel Principi.
Responsable de Ejecución	Juan Pablo Rumie Vittar.
Comentarios	s/c.

Tabla 2.7: Plantilla de Casos de Test 6

Nombre del caso de Test	Test 7
Descripción Abreviada	Visualización 3D expandir fuera del SD.
ID del caso de Test	7,0.
Descripción del caso de Test	Se debe verificar que se realiza correctamente la acción de expandir hacia otro monitor y contraer
	hacia el SD correctamente la visualización 3D.
Condiciones Previas	n/a.
Relaciones con otros casos de Test	n/a.
Pasos y condiciones de ejecución	Se ejecuta el SD, con la carga de un archivo que contiene varios vuelos o un vuelo, es indistinto y luego se comprueba presionando sobre la barra de reproducción el botón para extraer el componente visual 3D hacia otro monitor o pantalla y luego contraer, es decir, incorporarlo al SD de forma correcta en cualquier estado de funcionamiento del SD.
Resultados esperados	El SD debe realizar la expansión y contracción del componente visual 3D de forma correcta.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado • Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW en relación a la expansión del componente visual 3D hacia otra pantalla y el posterior regreso si así es deseable. Se realizo sin inconvenientes y visualizando en todo momento los movimientos de los objetos 3D (aeronaves para este caso).
Errores encontrados	n/a.
Responsable de Diseño	Juan Pablo Rumie Vittar.
Responsable de Ejecución	Juan Oviedo.
Comentarios	s/c.

Tabla 2.8: Plantilla de Casos de Test 7

Nombre del caso de Test	Test 9
Descripción Abreviada	Visualización de Elevación de Terreno.
ID del caso de Test	9,0.
Descripción del caso de Test	Verificar que cuando se realiza vuelos sobre territorio Argentino, en todo momento se visualice la elevación del terreno (se puede corroborar la misma con *Google Earth*) y únicamente cuando se realiza vuelos sobre el mar, el modelo 3D no este sobre la superficie marina, sino que debe estar a la Altitud de la aeronave.
Condiciones Previas	n/a.
Relaciones con otros casos de Test	n/a.
Pasos y condiciones de ejecución	Se ejecuta el SD, con la carga de un archivo que contiene varios vuelos o un vuelo, es indistinto y luego se comprueba en el componente 2D, e el cuadro de dialogo de la aeronave con información de parámetros de vuelo, el valor del parámetro *Elevation*.
Resultados esperados	El SD debe mostrar en su componente visual 2D el valor a la elevación correspondiente a su Latitud y Longitud, contrastada con, por ejemplo, *Google Earth*.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado • Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW en relación a la vista en el componente visual 2D el dato de elevación de terreno a lo largo de toda la reproducción de el o los vuelos, se visualiza correctamente en incluso pasajes a la Base Marambio en donde sobre el mar no hay terreno y luego se visualiza correctamente (nuevamente) sobre la Base Marambio.
Errores encontrados	n/a.
Responsable de Diseño	Ariel Principi.
Responsable de Ejecución	Juan Pablo Rumie Vittar.
Comentarios	s/c.

Tabla 2.9: Plantilla de Casos de Test 9

Nombre del caso de Test	Test 10
Descripción Abreviada	Generador de Gráficos.
ID del caso de Test	10,0.
Descripción del caso de Test	Verificar que el Generador de Gráficos realiza el gráfico correctamente de la o las variable y/o parámetros de vuelos escogidos.
Condiciones Previas	n/a.
Relaciones con otros casos de Test	n/a.
Pasos y condiciones de ejecución	Se ejecuta el SD, con la carga de un archivo que contiene varios vuelos o un vuelo, es indistinto. Se realiza completamente la reproducción hasta finalizar. Se ejecuta el Generador de Gráficos desde el menú principal y luego se comprueba la gráfica realizada por el Generador de Gráficos, se exportan a un archivo de imágenes el resultado.
Resultados esperados	El Generador de Gráficos debe mostrar correctamente y completamente el gráfico del o los parámetros de vuelos escogidos a visualizar.
Estado del caso de Test	• Ejecutado • Exitoso • Fallido • Frenado • Pendiente de ejecución • En construcción
Resultados Obtenidos	Resultado obtenidos del SW en relación a la creación de gráficos a partir de parámetros de vuelos seleccionados, se realizo correctamente exportándolos a imágenes para su posterior análisis sobre el o los vuelos.
Errores encontrados	n/a.
Responsable de Diseño	Juan Pablo Rumie Vittar.
Responsable de Ejecución	Juan Oviedo.
Comentarios	s/c.

Tabla 2.10: Plantilla de Casos de Test 10

Capítulo 3

Conclusión

El SD, como se especificó a lo largo del documento, tiene la característica de modularidad e interconexión de los diferentes componentes (característica del Proceso Unificado), lo que permite "desacoplar" componentes de software existentes y/o acoplar nuevos componentes de SW previamente desarrollados para ofrecer nuevas funcionalidades solicitadas y de esta manera, potenciar aún más el producto (todo esto es parte del ciclo de vida del proyecto a través de la metodología del Proceso Unificado (Ivar Jacobson y Rumbaugh, 1999)). Estas características de modularidad y de conectividad, permitieron no solo facilitar el desarrollo de nuevos requerimientos sobre nuevos avances, sino también la realización ordenada y sistemática de los *"ensayos por componentes"* y los ensayos *"de integración"* de partes del SD como, así también, de todo el SD.
Cabe destacar que a pesar de diferentes inconvenientes en la ejecución del proyecto, el mismo se ha cumplido en tiempo y forma con el cronograma previsto en el capítulo 1 y en cuanto a la factibilidad económica estipulada en la misma sección, solo hubo un desfasaje en materia de licencias SW a adquirir debido a retrasos en los tiempos para la aprobación del presupuesto.

3.1. Análisis de Metodología aplicada al Software

La aplicación de la metodología del Proceso Unificado (Ivar Jacobson y Rumbaugh, 1999) bajo la utilización del lenguaje de modelado UML (Rumbaugh y Booch,

2004) en el desarrollo del Software SD ha proporcionado una estructura clara y visual para el proceso de desarrollo del proyecto. Esto agilizó la comprensión de cada una de las fases o etapas y las relaciones existentes e intrínsecamente ligadas entre ellas, lo que es vital tanto para el equipo de desarrollo como para los usuario finales. La metodología del Proceso Unificado se puede analizar de cierta forma: observando las interfaces y conexiones entre cada etapa o fase que este debida y fuertemente ligadas entre ellas. Esto es importante cuando se realiza una regresión *"hacia atrás"* de alguna de ellas, por citar un ejemplo, si ocurre que en la etapa de diseño se observa que falta desarrollos de ciertos artefactos de diseño, esto se debe que en las etapas de casos de uso y análisis no se especificaron debidamente y completamente los artefactos correspondientes que demandan los requerimientos, con lo cual, para resolverlos *"se regresa hacia atrás"*, hasta los requerimientos técnicos y de allí, se agregan (*"hacia adelante"*) los correspondientes artefactos de casos de uso y análisis para cumplimentar con las correspondientes interfaces de los artefactos de diseño. Finalizado esto, se vuelve a *repasar*, y si lo requiere rehacer los artefactos de diseño, implementación y casos de test para que contemplen los agregados en las etapas anteriores, de allí, iterativo (se navega *"hacia atrás y hacia adelante"*) e incremental porque las actualizaciones contribuyen a la robustez, fiabilidad y completitud con los requerimientos técnicos estipulados a lo largo de todo el proyecto.
Los requerimientos técnicos o denominados *requerimientos funcionales* deben estar debida y completamente redactados en donde debe tomarse el tiempo suficiente para eliminar todo tipo de *ambigüedad* con respecto a los requisitos planteados por el cliente o usuario final, y evitar así a futuro rehacer gran parte de de los artefactos de cada una de las etapas que estipula la metodología de Proceso Unificado y evitar retrasos, costos innecesarios. Por último, esta metodología tiene como objetivo primordial garantizar que el SD cumpla con los requisitos especificados como se mencionó con anterioridad, lo que contribuye al aseguramiento de calidad del producto final.

3.2. Contribuciones

En la realización de desarrollos de productos de SW a nivel nacional en varios casos la tendencia principal es comparar estos con productos en el exterior con características similares (en contra partida al desarrollo nacional) y en muchos casos, esto incumbe a desembolsar sumas (por allí superior al desarrollo local) para tener *'algo ya listo y rápido'* por así decirlo dejando de lado la posibilidad y oportunidad del desarrollo Nacional, Federal y Genuino.
El desarrollo de este proyecto de software en la industria de la defensa nacional tiene contribuciones significativas y multifacéticas. Al implementar soluciones tecnológicas avanzadas, se mejora la capacidad operativa y estratégica de las fuerzas armadas, aumentando la eficiencia y precisión en las misiones. La incorporación de software de última generación fortalece la seguridad y la protección de la información crítica, permitiendo una mejor gestión de datos y comunicación en tiempo real. Además, el proyecto fomenta la innovación tecnológica y el desarrollo de capacidades nacionales, reduciendo la dependencia de tecnologías extranjeras y potenciando la autosuficiencia. Estas mejoras no solo optimizan las operaciones de defensa, sino que también impulsan el crecimiento económico y tecnológico del país, consolidando su posición en el ámbito global de la defensa.

3.3. Actualizaciones a Futuros

Con respecto a la primer versión desarrollada (RC) y sometida a procesos de Homologación por la DEyH de la FAA, surgen algunas cuestiones que los usuarios finales realizaron su apreciación y juicio crítico con respecto a mejoras en materia de innovación del SD. Las propuestas se detallan a continuación:

- Agregado de reproducción de vídeos (en caso que el SARM lo provea).
- Relieve del terreno en el componente 3D.
- Paleta de Herramientas con más contenido y adaptada a los requerimientos de cada usuario.

- Posibilidad de agregar modelos de aeronaves en el 3D mediante Plugins que se desarrollen en el momento que se necesiten, evitando así el desarrollo de un nuevo SD en el futuro.
- Reproducción de objetos de tipo aéreos, terrestres y marítimos con la posibilidad de mezclar diferentes tipos de objetos.
- Vista del componente 3D en forma FPV (*"First Person View"*).
- En el GV disponer de un *Tooltips* sobre los vuelos que con el arrastre del mouse muestre la fecha en la que se realizo cada vuelo.
- Factibilidad de incorporación al SD los datos (SARM) de los nuevos F-16, adquiridos por la FAA.
- Cálculo de la Desviación Magnética que es una derivación entre el Rumbo Geográfico y el Rumbo Magnético (Real).
- Posibilidad de agregar en el componente 3D, archivos KML generados con Google Earth[1] sobre objetos y/o *"waypoints"* de interés del usuario.

Estas son las principales y más importantes actualizaciones a realizar en el SD ya entregado (al momento de escribir este documento) a las diferentes brigadas aéreas y otros destinos, obviamente mientras más se utilice por los usuarios surgirán nuevas solicitudes. Todas serán estudiadas, analizadas y se les asignará una prioridad para su desarrollo, obviamente una vez que el SD actual se logre la Homologación al 100 %.

3.4. Videos Tutoriales

A continuación y como complemento demostrativo del SD, se especificarán los enlaces de Vídeos Tutoriales que se realizaron (como trabajo extra) a los usuarios, como material complementario a los manuales de usuario entregados cuyo objetivo es agilizar y facilitar la comprensión y utilización del SD.

[1] https://es.wikipedia.org/wiki/Google_Earth

- https://drive.google.com/file/d/1HlTWdzKUto9v0MT_tzRLaxTTu-i3MecA/view?usp=drive_link
- https://drive.google.com/file/d/1DuNus15-7YUvANlZ8sVDECl_MVvD6IYQ/view?usp=drive_link

Se cuentan con más vídeos del estilo, pero solo estos son los únicos que se pueden ser mostrados públicamente.

Bibliografía

IEC 61508, I. (2010). *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*. International Electrotechnical Commission (IEC).

IEEE 830, I. (2008). *Especificación de Requisitos según el estándar de IEEE 830*. Institute of Electrical; Electronics Engineers.

IEEE/EIA 12207, I. (2017). *Standard for Information Technology*. Institute of Electrical; Electronics Engineers.

Ivar Jacobson, G. B. & Rumbaugh, J. (1999). https://es.wikipedia.org/wiki/Proceso_unificado

RTCA DO-178C, R. I. (2011). *Software Considerations in Airborne Systems and Equipment Certification*. Software Integrity Assurance Considerations for Communication, Navigation, Surveillance; Air Traffic Management (CNS/ATM) Systems.

RTCA DO-278, R. I. (2011). *Software Standard for Non-Airborne Systems*. Software Integrity Assurance Considerations for Communication, Navigation, Surveillance; Air Traffic Management (CNS/ATM) Systems.

Rumbaugh, J. & Booch. (2004). https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

SAE ARP 4754B, S. (2023). *Guidelines for Development of Civil Aircraft and Systems*. SAE International.

SAE ARP 4761A, S. (2023). *Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment*. SAE International.

teams at U.S. military. (1990). https://www.nga.mil/