Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FIN DE CARRERA

TÍTULO DEL TFC: Estación de seguimiento SKY-EYE para UAVs: integración visual de componentes de seguimiento y georeferenciación sobre GIS

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Sistemas de Telecomunicación

AUTOR: Juan Manuel Lema Rosas

DIRECTOR: Marco A. Peña Basurto

FECHA: 9 de julio de 2007

Título: Estación de seguimiento SKY-EYE para UAVs: integración visual de componentes de seguimiento y georeferenciación sobre GIS

Autor: Juan Manuel Lema Rosas

Director: Marco A. Peña Basurto

Fecha: 9 de julio de 2007

Resumen

Gracias a los grandes avances en la aeronáutica y en las telecomunicaciones, hoy en día se puede conseguir, por raro que parezca, que unos aviones sin tripulación alguna y sin ser teledirigidos vuelen por la superficie de la tierra. Es decir, por sí mismos. A estos aviones se los conoce como UAV, acrónimo proveniente del inglés que significa vehículos aéreos no tripulados.

Esta tecnología ya está desarrollada. Incluso se utiliza hace ya años con fines militares, pero no en el mundo civil.

Es de gran interés aprovechar esta tecnología y así, desarrollándola un poco más, conseguir dar a estos aviones nuevas utilidades para que puedan ser aprovechadas por la sociedad.

Un posible uso es el de detectar o prevenir incendios forestales. Los aviones volarán por las zonas de más calentamiento de la tierra, y al detectar focos de calor, enviarán un mensaje de alarma para que se actúe de inmediato.

Por tanto el objetivo de este proyecto es contribuir con el desarrollo de las nuevas utilidades del UAV. En concreto desarrollar una aplicación informática para la estación de tierra que visualice la ruta que seguirá el avión, y una vez volando recoja y administre toda la información que el UAV pueda enviar.

Title: Estación de seguimiento SKY-EYE para UAVs: integración visual de componentes de seguimiento y georeferenciación sobre GIS

Author: Juan Manuel Lema Rosas

Director: Marco A. Peña Basurto

Date: July, 9th 2007

Overview

Thanks to the great improvements in the aeronautic and telecommunication sectors, nowadays we can achieve to fly some air planes without the human help, just by their own. These air planes are known as UAV, meaning Unmanned Aerial Vehicle.

This technology is already developed and used by the military for some years, but not at the civil world.

It is very interesting to take advantage of this technology, study more about it and develop new applications in order to use them to solve problems in the civil society.

One of these problems could be the fire in our forests. The planes may fly over the conflicting zones, and when they detect some fire focus, they may send a message alarm in order to react immediately.

Thus, the main objective of this project is to contribute with the development of these new applications of the UAV. To be more specific the aim is to develop the software located at the ground station. This software has to show the flight plan and all the information received from the plane.

ÍNDICE

INTRODUCCIÓN

Gracias a los grandes avances en la aeronáutica y en las telecomunicaciones, hoy en día se puede conseguir, por raro que parezca, que unos aviones sin tripulación alguna y sin ser teledirigidos vuelen por la superficie de la tierra. Es decir, por sí mismos. Estos aviones se les conocen por UAV, acrónimo proveniente del inglés que significa vehículos aéreos no tripulados.

Esta tecnología ya está desarrollada. Incluso se utiliza hace ya años con fines militares, pero no en el mundo civil.

Es de gran interés aprovechar esta tecnología y así, desarrollándola un poco más, conseguir dar a estos aviones nuevas utilidades para que puedan ser aprovechadas por la sociedad.

Un posible uso es el de detectar o prevenir incendios forestales. Los aviones volarán por las zonas de más calentamiento de la tierra, y al detectar focos de calor, enviarán un mensaje de alarma para que se actúe de inmediato.

Por tanto el objetivo de este proyecto es contribuir con el desarrollo de las nuevas utilidades del UAV. En concreto desarrollar una aplicación para la estación de tierra que visualice la ruta que seguirá el avión, y una vez volando recoja y administre toda la información que el UAV pueda enviar.

Cabe destacar que para desarrollar esta aplicación han sido esenciales las asignaturas de LPII y DSEM, en las que se introduce al alumno en el mundo de la programación con objetos y desarrollo de pequeños proyectos.

La planificación inicial del proyecto parte como la de todos los TFC, una duración de cuatro meses. Debido a demoras en el análisis del problema inicial, y problemas con algunas herramientas de desarrollo el proyecto se ha alargado hasta llegar a los cinco meses.

La distribución de capítulos se ha hecho igual que cualquier proyecto de software. Se comienza con un estudio del contexto tecnológico y análisis del problema, luego se especifica, diseña e implementa. Se acaba con el balance y las conclusiones.

En el primer capítulo se sitúa el problema, se analiza el contexto general y se realiza una primera planificación.

Ya en el segundo capítulo se realiza un análisis tecnológico detallado de todo lo que tenga que ver con el proyecto. Se especifican los problemas concretos.

En el capítulo de especificación, se da una solución técnica al problema. En este capítulo se hace una nueva planificación teniendo en cuenta las etapas ya pasadas, y realizando un pronóstico un poco más detallado de las siguientes.

El capitulo de diseño explica la puesta en marcha de la aplicación. Se habla de las herramientas de desarrollo y las estructuras de clases.

Ya en implementación se comentan los aspectos más interesantes a la hora de construir la aplicación. Se comentan las pruebas realizadas tras la integración.

El capítulo de balance explica los principales problemas a la hora de desarrollar el proyecto. Se hace una revisión de objetivos, y una revisión de gastos. Se realiza un diagrama de duración de tareas final con el fin de poder contrastar con el inicial.

Finalmente el capítulo de conclusiones agrupa todos los aspectos que han sido de importancia para el proyecto. Se habla con más detalle del cumplimiento de objetivos, modos de trabajo en grupo, etc.

Se puede encontrar un capítulo de anexos en el que se vuelca toda la información relacionada con el trabajo de estos cinco meses. Será de gran ayuda para que una vez leído se puedan contrastar datos e investigar un poco más el tema.

CAPÍTULO 1. CONTEXTO TECNOLÓGICO Y ANÁLISIS DE OBJETIVOS

1.1. Motivación: antecedentes, y motivación personal

El proyecto SKY-EYE nace en la EPSC, junto con la colaboración de otras empresas y universidades, con el fin de diseñar una nueva herramienta basada en aviones UAV. La idea es volar una serie de aviones no tripulados, que estos desempeñen una misión sobre el terreno, y enviar esta información a tierra, para que un operario pueda tomar decisiones. Una posible aplicación es el control de incendios. A veces es complicado llegar a los focos de calor por tierra. Con este sistema de aviones UAV, conseguiremos obtener información al momento y sin peligro.

Es un proyecto de aproximadamente dos años de duración en el que se tocan diversos temas relacionados con la ingeniería en telecomunicaciones: desarrollo del sistema embarcado, diseño y desarrollo de las aplicaciones de estación de tierra, comunicaciones entre el avión y tierra, entre otros. Para ello el proyecto se divide en distintos grupos que realizan distintas tareas. La EPSC ha creado el grupo ICARUS, un conjunto de objetivos y personas que realizarán una parte de SKY-EYE. Otro de los grupos importantes dentro de SKY-EYE es la empresa INFINIFISH que se encargará de coordinar el proyecto, y desarrollar aplicaciones.

Teniendo en cuenta la envergadura del proyecto, y coincidiendo con el final de mi carrera, me ha parecido interesante unirme al equipo de investigación y desarrollo de INFINIFISH con el fin de realizar mi TFC.

De esta manera comienza una etapa de investigación, documentación, diseño, desarrollo y comprobación con todo lo que tenga que ver con la aplicación de la estación de tierra.

1.2. Contexto y oportunidad

En el la escuela EPSC se encuentran, entre otras, las facultades de ingeniería técnica de Telecomunicación (Sistemas de Telecomunicación y Telemática) y Aeronáutica. Si se tiene en cuenta el interés por la investigación y desarrollo que esta universidad ha adquirido, se puede decir que es el mejor contexto para un proyecto de las características de SKY-EYE ya que se dispondrán de muchos expertos en la materia, y proyectistas que desempeñen tareas diversas.

Por otro lado, en Cataluña, y en particular Barcelona, se tiene un gran interés en proyectos de aeronáutica. Hay que tener en cuenta que la idea de volar aviones no tripulados puede no ser del agrado de los representantes políticos al principio, por lo que puede llegar a ser un problema. Por otro lado estos

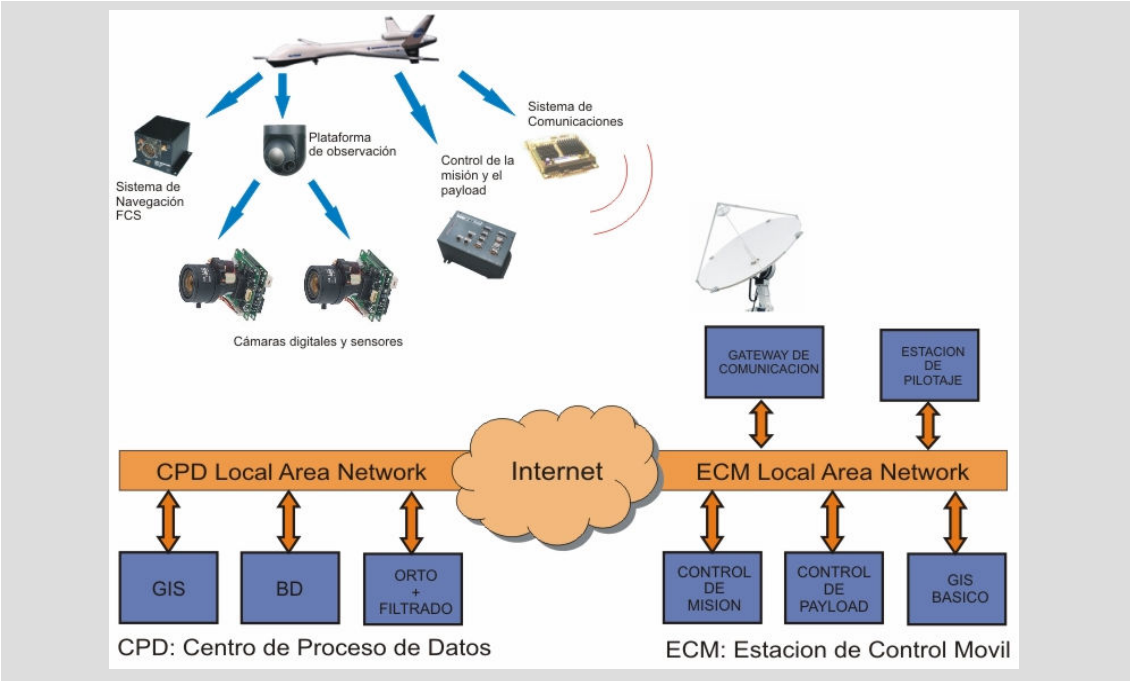
aviones son utilizados desde hace años con fines militares, por lo que la tecnología ya está desarrollada. De esta manera, con poca inversión se pueden conseguir buenos objetivos.

1.3. Escenario

El proyecto SKY-EYE trata de estudiar y desarrollar un sistema embarcado en un avión no tripulado (UAV), en el sistema embarcado irán módulos de control de vuelo, de misión y toda la carga de pago de captación de información (cámaras fotográficas en espectro visible o infrarrojo, sensores, etc.). El proyecto SKY-EYE podría llegar a tener muchas aplicaciones posibles, control y vigilancia de grandes infraestructuras, oleoductos, red viaria o ferroviaria, salvamento y vigilancia marítima o forestal, etc. Aunque a corto plazo la aplicación principal en la que se centrará el proyecto será la detección, control y análisis en tiempo real de incendios forestales, como se ha dicho esta es solo una de las aplicaciones posibles del sistema.

El esquema general del proyecto sería el siguiente (Figura 1):

Figura 1. Esquema de Estaciones Base



Uno o varios UAV, con un sistema a bordo, conectados y operando alrededor de una estación de control. Una estación central de seguimiento, conectadas a las diversas estaciones de control, con una aplicación creada a medida para el proyecto.

Al comienzo del proyecto, controlaremos y seguiremos a los UAV desde una misma estación sin tener la necesidad de trabajar con redes TCP/IP. No obstante, para dar una visión más amplia de cara a la extensión y ampliación, pensamos en este esquema. Una vez que consigamos un funcionamiento

correcto, el separar dichas estaciones nos servirá para poder conectar varias estaciones de control (unidades móviles) y que estén conectadas a través de la red a una estación fija.

1.4. Tecnología

1.4.1. UAV (Unmaned Aerial Vehicle)

La plataforma embarcada en el UAV constará de un sistema de control de vuelo y de un sistema Hardware-Software. Este sistema se dedicará a controlar el plan de vuelo (trayectorias y waypoints, maniobras de escape), y de controlar la misión con todas las acciones sobre el payload, ya sean cámaras de fotografía, cámaras de video, sensores de IR, sensores térmicos, etc.

1.4.2. Estación de control

Esta estación de control se situará en el área de operaciones de los UAV. Constará de dos sistemas de comunicación con el UAV. A través de un radioenlace (ancho de banda bajo) se enviará toda la información referente a la telemetría proporcionada por el sistema de control de vuelo y además el estado de los equipos.

El segundo sistema de comunicación será de mayor ancho de banda basado en una tecnología aún por definir (Wi-Fi, WiMax). Con este sistema de comunicación podremos obtener todos los datos del plan de vuelo y toda la información obtenida de la misión.

El primer sistema de comunicación nos interesa que esté conectado en todo momento para poder situar y controlar el UAV. En el segundo, al necesitar un ancho de banda muy superior, no disponemos de una tecnología que nos asegure la conexión en todo momento, así que debido a falta de cobertura se podrá interrumpir la conexión. Por ello deberemos tomar algunas medidas.

En caso de que se corte la comunicación, cuando se retome no necesariamente se seguirá por donde se cortó, sino por donde se encuentra en ese momento, almacenando la información no transmitida. Esta información almacenada se podrá transmitir en intervalos en que la misión se encuentre inoperativa, o una vez el UAV haya aterrizado.

1.4.3. Estación de seguimiento

Se podrá situar en cualquier espacio del territorio, conectándose a las diversas estaciones de control a través de una red de comunicación convencional. Aquí

se podrá coordinar todas las estaciones de control, todos los UAV con sus planes de vuelo y sus misiones.

Para esta estación se diseñará una aplicación de software con diferentes funciones. La principal función será la de un sistema GIS, con cartografías y modelos de elevación y capaz de georeferenciar y ortonormalizar la información de la misión. También tendremos las funciones de un planificador de vuelo y misión, para poder diseñar el recorrido, waypoints y maniobras, diseñar también la misión indicando las acciones del payload, poder hacer un seguimiento en tiempo real de la telemetría, estado del UAV y de la información de la misión obtenida hasta el momento.

1.5. Objetivos

1.5.1. Objetivos Iniciales

El objetivo más inmediato para el proyecto SKY-EYE es la creación de un prototipo que integre todo el conjunto de Sistema Abordo, Estación Control, Estación Seguimiento. Este prototipo se presentará en la FireParadox (ver http://www.fireparadox.org/) de Agosto de este año en Portugal.

Los objetivos para la realización de este prototipo serán tres, uno para cada parte del conjunto:

- Diseñar un sistema para embarcar en un UAV capaz de capturar todo tipo de información a través de cámaras fotográficas, de video, sensores de la superficie terrestre.
- Diseñar los sistemas de transmisión de datos entre los UAV y las estaciones de control.
- Diseñar una estación de seguimiento con la aplicación de software apropiada para controlar todo el sistema.

Del desarrollo del sistema de embarcado en el UAV y de las comunicaciones a tierra se encargaran el grupo de investigación ICARUS en la EPSC, y del diseño de la estación de seguimiento y de la aplicación de software, respaldada por sistemas GIS se encargará la empresa INFINIFISH.

1.5.2. Objetivos Grupo Infinifish

Los objetivos del grupo Infinifish los intentarán llevar a cabo los proyectistas Borja López, Miriam González, y Juan Manuel Lema, con la colaboración de miembros de dicha empresa. Podemos dividir los objetivos en estas tres tareas:

- Georeferenciación y ortonormalización de imágenes. Esta tarea estará centrada en las capas de lógica de negocio y la de datos, ya que deberá implementar toda la algorítmica de georeferenciación que se usa en GIS.

- Plan de vuelo. Este TFC se encargará de la lectura de planes de vuelo XML, de crear un modelo conceptual acorde, de la gestión de imágenes asociadas, etc. Por ello estará centrado en las capas de datos y de lógica de negocio. En relación a un GIS, se encargará de la gestión de coordenadas de cartografías.
- Interfaz de usuario. Se trata de diseñar la interfaz de usuario, centrándose en la capa de presentación y con acceso a la cada de gestión de la aplicación. Este diseño deberá tener en cuenta el aspecto visual de los GIS y de parte de un control de vuelo.

Se repartirán estos temas entre los proyectistas. En este TFC se explicará el desarrollo de la última tarea.

1.5.3. Objetivos Individuales

Para poder diseñar una aplicación en la que se visualice terreno, y se pinte sobre él, es conveniente conocer un poco más a fondo los sistemas de información geográfica (GIS). De esta manera un primer objetivo será dicho estudio. Una vez familiarizado con el sistema comienza la parte de diseño y desarrollo.

El principal problema que se plantea, es interpretar la realidad que se observa desde el avión en unas aplicaciones que distribuyan correctamente la información para visualizarla de la forma más lógica y útil posible.
La interfaz de usuario debe servir para volcar toda la información que creamos referente al plan de vuelo, y toda la información que recibimos de los paquetes de telemetría. Se deberá también desarrollar las herramientas necesarias para un buen control del mapa como por ejemplo el zoom.
Como es de esperar en toda GUI, tendrá que estar muy bien estructurada con el fin de representar gran cantidad de información y poder discernirla correctamente

Se debe tener en cuenta que esta parte del proyecto es en la que se debe mostrar todo el trabajo necesario que hace el programa, es decir, los otros dos TFC, y es de gran importancia plasmar esto de una forma simple. Por lo tanto se debe analizar el modelo conceptual y las estructuras de las librerías creadas, interactuar con ellas, y dar esta información al usuario.

Además de los objetivos técnicos del proyecto, existen otros tipos de objetivos no menos importantes. El primero de ellos es crear un TFC de interés, con buenos resultados, que sea el principio de una serie de investigaciones que se sigan desarrollando al correr del tiempo. Al margen del trabajo final de carrera, están los intereses propios de la persona. Los míos son conocer desde un poco más cerca el mundo profesional de un ingeniero. Ese ha sido el motivo principal para investigar fuera de ICARUS y las instalaciones que la EPSC ofrece. Así se consigue experiencia laboral, no sólo en el desarrollo del proyecto, sino también en el trabajo en equipo, la relación profesional con los compañeros, el cumplimiento de horarios, etc.

La combinación de mis objetivos personales, junto a los del proyecto permitirá que la tarea del desarrollo del TFC no sea simplemente un trabajo de investigación, y documentación, sino una buena experiencia tanto académica como profesional.

1.6. Planificación y estimación de costes

Un proyecto de esta envergadura requerirá tiempos, esfuerzos y material. En este apartado se detallará el tiempo y dinero que requerirá el desarrollo del proyecto.

1.6.1. Planificación

En un proyecto de software se requiere una etapa previa de investigación. En esta etapa se analiza el problema y se resuelven los materiales y tiempo requerido. Una vez ubicado el problema se entra en una etapa de especificación en la que se da una solución técnica. Se realiza una planificación detallada para pasar al periodo de diseño e implementación donde se resuelve de forma práctica el problema. Una vez acabadas estas etapas se entra en el periodo de documentación del trabajo realizado.

La Figura 2 muestra una planificación previa al proyecto. En el capitulo de Balances se explicarán las diferencias debidas a cambios de objetivos y problemas sufridos.

Figura 2. Esquema de Estaciones Base

Nombre de tarea	Duración	Comienzo	Fin
1- Estudio y comprensión del problema	18 días?	mié 2/7/07	vie 3/2/07
2- Análisis detallado del estado del arte	23 días?	mié 2/14/07	vie 3/16/07
3- Evaluación de sistemas GIS	18 días?	mié 2/28/07	vie 3/23/07
4- Especificación y Plan para las fases posteriores	18 días?	mié 3/7/07	vie 3/30/07
5- Georeferenciación de información	19 días?	jue 3/15/07	mar 4/10/07
6- Diseño del lector de plan de vuelo	19 días?	jue 3/15/07	mar 4/10/07
7- Diseño del GUI de la estación de seguimiento con GIS	19 días?	jue 3/15/07	mar 4/10/07
8- Desarrollo de componentes	28 días?	mié 4/11/07	vie 5/18/07
9- Integración del sistema	13 días?	mié 5/2/07	vie 5/18/07
10- Documentación TFC	17 días?	mar 5/15/07	mié 6/6/07

Esta planificación se ha creado a partir de las etapas explicadas anteriormente de un proyecto de software.

- **Estudio y comprensión del problema:** Esta etapa es el primer contacto con el proyecto. Se analiza de forma general los temas relacionados con SKY-EYE como son el plan de vuelo y de misión, GIS o estaciones de seguimiento.

- **Análisis detallado del estado del arte:** Análisis general de problema y definición de objetivos. Duración una semana.

- **Evaluación de sistemas GIS:** Etapa de documentación relacionada con los GIS. Duración dos semanas.

- **Especificación y Plan para fases posteriores:** Una vez analizado el problema y el entorno se detallarán los objetivos y la planificación. Duración una semana.

- **Diseño, implementación y pruebas:** Es la etapa en la que se trabajará por separado:
  - Gestión de Plan de Vuelo
  - Georeferenciación y Ortonormalización de información
  - Diseño del GUI de la estación de seguimiento integrado con GIS.

  Esta etapa durará un mes.

- **Desarrollo de componentes**: Una vez acabado el trabajo en paralelo se trabajará conjuntamente para acabar de desarrollar las etapas. Duración de una semana.

- **Integración del sistema:** Se agruparán las tres vías de trabajo en una sola. Se deben realizar pruebas de integración. Dos semanas de duración.

- **Documentación del TFC:** Se recopila toda la información creada para luego ser documentada. En esta etapa también se trabaja en paralelo para conseguir tres TFC. Duración de tres semanas.

En todas las tareas se realizará o bien un documento, o una aplicación. Estos documentos serán muy útiles a la hora de redactar la memoria. Una vez acabada la fase, se entregan los documentos a los responsables de INFINIFISH para que supervisen el trabajo. Se deberá realizar cambios o correcciones aun estando en otras etapas.

1.6.2. Estimación de Costes

Además de hacer una previsión de tiempo, es interesante realizar otra de costes económicos. Existen dos tipos de costes, los que se destinan a sueldos, y a material. En este proyecto hay que tener en cuenta que hay tres miembros.

- ***Recursos humanos***

Los gastos irán en función de la tarea que se este desempeñando ya que no cuesta lo mismo la hora del informático que la del documentador.

El proyecto está compuesto por tareas que realizan tres perfiles profesionales distintos: Analista informático, Programador Informático y Documentador.
El Analista informático se encargará de las siguientes tareas:

- Estudio y comprensión del problema
- Análisis detallado del estado del arte
- Evaluación de sistemas GIS
- Especificación y planificación para las fases posteriores

Estas cuatro tareas ocupan 26 días laborables que van del día 7 de febrero al día 14 de marzo.

El Programador Informático se encargará de las tareas:
- Diseño, implementación y pruebas
- Desarrollo de componentes
- Integración del sistema

Las tres tareas se desarrollan en 47 días laborables desde el día 15 de marzo al día 18 de mayo.

El documentador se encargará de la tarea:
- Documentación TFC

La última tarea se inicia el día 15 de mayo y acaba el 6 de junio, sumando 16 días laborables.

- ***Infraestructura***

El material necesario para el desarrollo de esta aplicación supone un gasto que se debe tener en cuenta a la hora de la planificación.

Para cada miembro del grupo se requiere un ordenador con conexión a Internet para la investigación y desarrollo del proyecto. Se estima un precio de 1000€ por terminal.

Este ordenador debe disponer del software Windows XP y Microsoft Office XP. Además se requerirá un software para el desarrollo de sistemas GIS. Al no estar definido se debe asumir un incremento en el precio final.

Se deberá adquirir la licencia del Sharepoint, el portal en el que se vuelca toda la información desarrollada por cada miembro del grupo.

Se asumirán 500€ en gasto de material de oficina como por ejemplo cd, dvd, hojas, la compra de cartografías, etc.

- ***Coste total***

Para realizar el coste total se sumará el gasto en sueldos y los materiales.

Los salarios se calcularán contando una rutina de trabajo de 25 horas semanales. El grupo dispondrá de tres miembros que desarrollaran distintas tareas en función de en qué etapa del proyecto se encuentren. El sueldo por hora trabajada de cada rubro se estima de las remuneraciones medias que se indicaron en proyectos de final de carrera similares realizados en la EPSC y la FIB.

RECURSOS HUMANOS	Recursos	Días de trabajo	Horas por día	€/hora	Horas Recurso	Coste Recurso	Horas Totales	Coste Total
Analista Informático	3	26	5	32	130	4.160 €	390	12.480 €
Programador	3	47	5	25	235	5.875 €	705	17.625 €
Documentador	3	16	5	20	80	1.600 €	240	4.800 €
			TOTAL		**445**	**11.635 €**	**1335**	**34.905 €**

INFRAESTRUCTURA	Cantidad	coste unidad	coste total
Estación de Trabajo	3	1.000 €	3.000 €
Licencia Windows XP	3	159 €	477 €
Licencia Sharepoint	1	2.000 €	2.000 €
Licencia Microsoft Office	3	450 €	1.350 €
Licencia Entorno de Desarrollo	por determinar		
Licencia Librerías GIS	por determinar		
Materiales de oficina varios			500 €
		TOTAL	7.327 €
		COSTE TOTAL	**40.405 €**

1.7. Metodología de trabajo

El lugar de trabajo será en la propia empresa INFINIFISH. Se dispone de un ordenador por proyectista. Estos ordenadores serán cedidos por la EPSC, y la empresa completará con todo el resto de material necesario.

Habrán reuniones una vez por semana para comentar cómo ha ido evolucionando el trabajo, y resolver posibles dudas. Esta reunión de seguimiento tiene previsto celebrarse los martes a las 12.30 de la mañana. Para esta reunión se debe hacer un “Orden del Día”, en la que se detallen los temas a tratar, y al finalizar la reunión se deberá redactar un “Acta de Reunión” en el que se expliquen los temas tratados. Existirán otras reuniones de trabajo puntuales en caso de ser necesarias. Por ejemplo se deberán hacer reuniones con los miembros del equipo ICARUS, con el fin de sincronizar fechas de trabajo, ya que hay información que genera ICARUS, que es imprescindible para que INFINIFISH continúe trabajando.

Se hará un control de horas de trabajo. Se contabilizará todo el tiempo de trabajo, y se clasificará en función del tipo de trabajo realizado. Toda esta información será guardada en una hoja de cálculo que debe irse actualizando. Esta información será muy útil para realizar un balance final al terminar el proyecto.

Todos los documentos se irán subiendo en un portal. A este portal sólo tendrán acceso los proyectistas de INFINIFISH y colaboradores. De esta manera se tendrá un repositorio común a donde acceder y volcar información.

CAPÍTULO 2. ANÁLISIS DEL PROBLEMA

Antes de empezar a trabajar se debe investigar todos aquellos ámbitos que estén relacionados con el proyecto. Como ya se ha mencionado anteriormente un campo imprescindible es el de los sistemas de información geográfica o GIS (geographic information system). También será necesario buscar información de Planes de Vuelo y Estaciones de Seguimiento.

2.1. Sistemas de Información Geográfica

2.1.1. ¿Qué es un GIS?

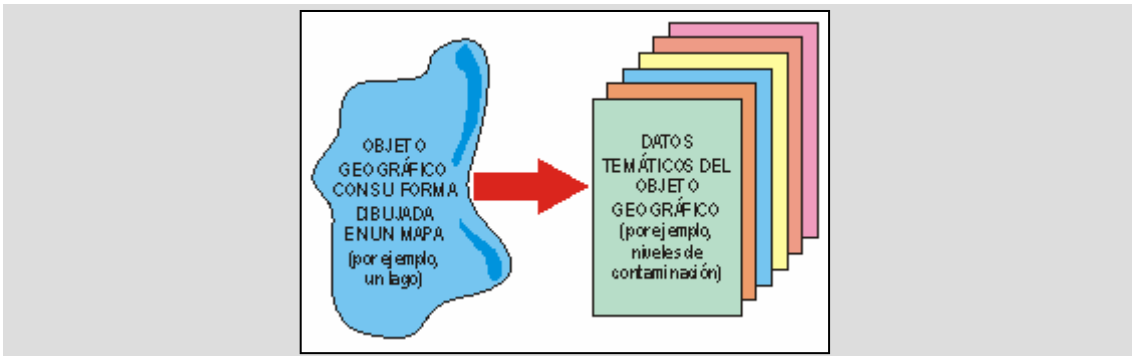
Un GIS (del inglés Geographic Information System) se encarga de administrar la información geográfica de forma eficaz y completa. De esta manera, a través de una base de datos de un sistema GIS se puede capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación, gestión, visualización, etc.

La idea nació en 1854 a manos del Dr. John Snow, cuando cartografió sobre un mapa los casos de cólera conocidos en la región de SoHo, Londres. De esta manera, mediante dichos puntos consiguió encontrar un pozo de agua contaminado, causante del brote.

Se puede ver el caso del Dr. John Snow como el primero de los GIS. De esta manera es más factible hacerse una idea que con un GIS se puede ubicar con precisión todos los lugares geográficos que tienen las características de interés y por lo tanto decidir dónde actuar.

Ésta no es la única propiedad de los GIS. Un GIS se caracteriza por dividir su información en capas. Están las capas con vertiente espacial y las de vertiente temática. De esta manera, para un lago en un GIS, no sólo se tendrá capa de profundidad y dimensiones (capas espaciales), sino que se puede tener grados de contaminación, niveles de flora y fauna, etc. (capas temáticas (Figura 3)).

Figura 3. Dibujo Explicativo de Sistemas GIS



De esta manera, a través de un GIS se consigue disponer de una base de datos muy completa con la que podremos trabajar de forma rápida y efectiva.

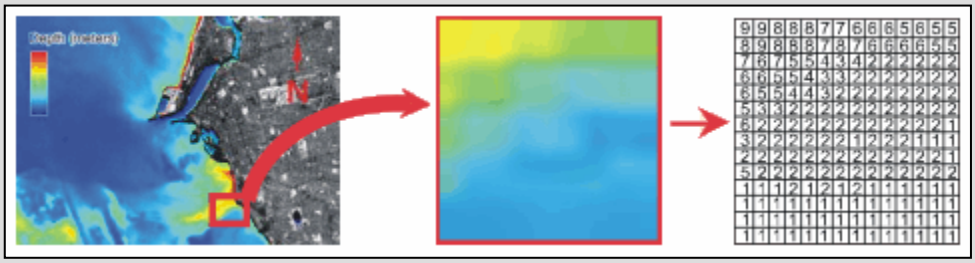
2.1.2. Modelos de GIS

Existen diferentes maneras de almacenar la información geográfica en las bases de datos. Entre ellos hay dos principales modelos: Raster y Vectorial

Modelo Raster

Este modelo divide la zona de interés en pequeñas celdas regulares llamadas píxeles. Estas celdas están en todo momento georeferenciadas ya que sus coordenadas son conocidas. A estas divisiones se les atribuye unos valores numéricos en función de su valor temático (Figura 4). Hay que tener en cuenta que a mayor numero de píxeles, mejor resolución, y mayor costo computacional. Por ello se debe llegar a un compromiso en función de los objetivos y de los equipos. El modelo raster se centra en las propiedades del espacio más que en la precisión de la localización.

Figura 4. Modelo Raster



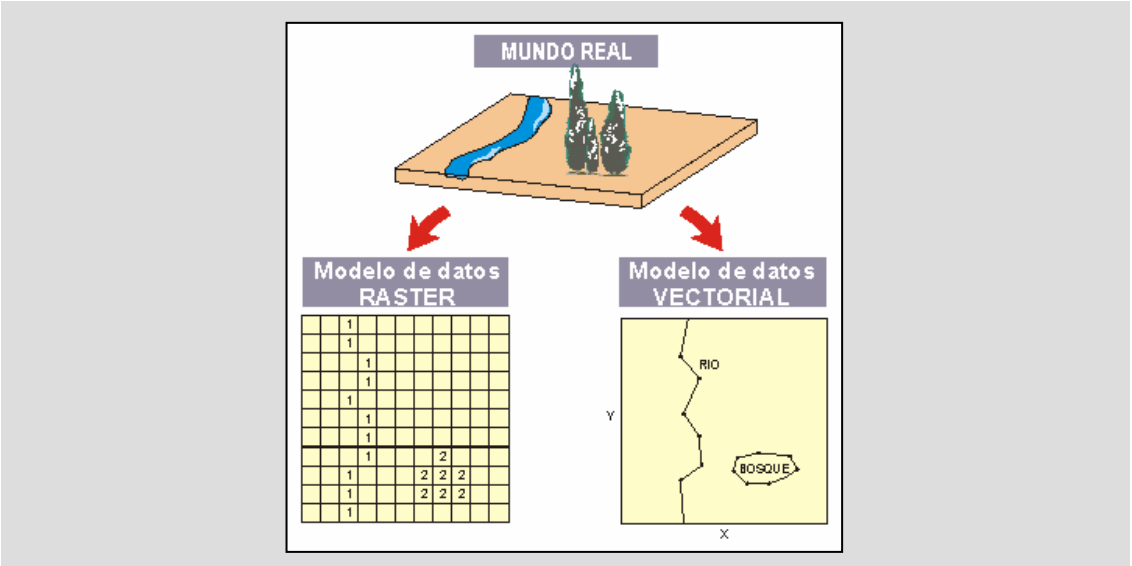
Modelo Vectorial

Son aquellos que utilizan herramientas gráficas para el almacenamiento e interpretación de la información. Un punto puede ser un pozo de agua, una línea una carretera, un circulo un río, etc. Se aprecia de inmediato la necesidad de utilizar un sistema de coordenadas adecuado que ayude a referenciar toda esta información. Existe el sistema de coordenadas geográfico (x,y,z), y el sistema de coordenadas UTM en el que se traza sobre la tierra líneas polo a polo con una separación entre ellas de seis grados. De ésta manera resultan 60 zonas UTM (husos). El interés de las representaciones vectoriales se centra en la precisión de localización de los elementos sobre el espacio.

En el esquema de la Figura 5, se puede ver la principal diferencia entre los modelos citados:

Los GIS vectoriales son más populares en el mercado. No obstante, los SIG raster son muy utilizados en estudios medioambientales donde se requiere una mayor precisión espacial (contaminación atmosférica, distribución de temperaturas, localización de especies marinas, análisis geológicos, etc.). Los software mínimamente completos pueden trabajar con ambos tipos.

Figura 5. Dibujo ilustrativo: Raster y Vectorial



2.1.3. Aplicaciones de Sistemas GIS

¿Para qué se necesita un sistema GIS? La respuesta es prácticamente evidente: Se requiere aprovechar al máximo las imágenes tomadas desde el UAV. No basta con sacar las fotos para que un operario las mire, sino que un sistema informático se encargue de abstraer toda la información posible de la foto y representarla adecuadamente.

Los pasos a seguir, a grandes rasgos serán los siguientes. Una vez obtenida la foto, el GIS deberá georeferenciar (ubicar esta foto en un cartograma), ortonormalizar (adaptar la foto para su correcta visualización), y almacenar toda la información en una base de datos.

De esta manera se podrá hacer consultas y trabajar sobre el sistema y base de datos con información actualizada.

Otra de las aplicaciones está relacionada con el seguimiento del avión y la misión. Interesa saber exactamente qué es lo que hace el avión en realidad, independiente del plan de vuelo. Este mecanismo también tendrá que ser visualizado en el mapa. Por lo tanto, para ambas aplicaciones se deberá interactuar con “capas” sobre la cartografía visualizada en ese momento.

2.2. Software y Hardware GIS

2.2.1. Información General

Existen muchos software GIS, tanto libres como no libres, de código abierto o de código cerrado. Se define libre a esos software en los que no se deba pagar

licencias para utilizarlos. Son de código abierto aquellos software que faciliten su código base.

Estos programas deben ser investigados a fondo para ver si se pueden aprovechar sus herramientas y así poder utilizarlas para los fines del proyecto. Para ello el, o los, software elegidos deben ser extensibles y/o programables. De esta manera, sin tener porqué ser de código abierto, se podrá programar utilizando sus funciones, o extenderlos.

El proyecto SKY-EYE no intentará crear un GIS. De hecho un GIS tiene muchas más aplicaciones de las mencionadas aquí. El objetivo es intentar sacar el máximo provecho a unas fotos tomadas en unos terrenos. Por ello, se intentará aprovechar, en la medida de lo posible, alguna herramienta que ya exista en el mercado, y así resolver el problema de tratamiento de foto y su integración con la base de datos.

2.2.2. Criterios de selección de GIS

Como se ha comentado anteriormente se precisa un sistema o librería GIS para poder cubrir una serie de necesidades presentes a la hora de afrontar el proyecto SKY-EYE. Por tanto se realiza una clasificación de las características más importantes a tener en cuenta en los GIS, a la hora de seleccionarlos como candidatos para llevar a cabo los requerimientos del proyecto. Las características tenidas en cuenta para su selección son las siguientes:

Gestión de BBDD: Se debe disponer de una base de datos para poder realizar búsquedas, insertar/eliminar información de los elementos, geográficos o no, que interesen. La idea es tener un amplio abanico de información sobre características geográficas, ya sea en tablas como en ficheros, para poder estudiar las imágenes tomadas desde el UAV. Pesará un 15% en la evaluación.

Lenguaje: La idea es encontrar un GIS en el cual se pueda modificar, alterar o insertar código si así se cree conveniente. Un lenguaje del cual se tenga conocimiento o se intuya que el aprendizaje es rápido y eficiente. Es un aspecto a tener en cuenta, pero no será determinante como para darle porcentaje de peso.

Código Abierto: Disponer de total libertad para modificar código. Tener acceso completo a la aplicación para poder realizar un estudio de la metodología creada para el desarrollo de las diversas funciones. Disponer de una aplicación de la cual no se tenga acceso al código no es de utilidad ya que no se necesita un sistema de información geográfica en sí, sino de una serie de herramientas que ya tienen creadas.

En caso contrario sería necesario tener opción a una API que exponga la funcionalidad desde el punto de vista de la programación para así tener opción a implementar nuestro propio código. Se considera este aspecto muy importante, por ello pesará un 40%.

Georeferenciación/Ortonormalización: Una vez se disponga de información capturada mediante la misión del avión, se tendrá que disponer de una herramienta capaz de georeferenciarla, disponer de información sobre elementos cercanos a esa posición y de diversas características de esas coordenadas.
Respecto a la ortonormalización la idea es disponer de una herramienta capaz de modificar las imágenes tomadas desde el UAV para así tenerlas lo más normalizadas posible. Dependiendo del ángulo con el que se tomen estas imágenes se tendrá una calida superior a la hora de tratar esta información. Se requiere que el tener siempre la mejor calidad posible de la información no sea un problema, ni tarea a realizar. No es un aspecto muy importante, ya que es una tarea a implementar. Pesará un 5%.

Formato: El formato de lectura/escritura de la información no debe suponer un problema de desarrollo, conversión o tiempo. Se pretende tener diversidad de formatos, diversidad a la hora de tratar la información. Si se dispone de un extenso abanico de posibilidades de lectura de los archivos, la información será procesada sin que con ello se tenga una gran perdida de tiempo debido a conversión de formatos. Tendrá un 20% de peso.

Licencias: En algunos GIS, el poder tener acceso completo a su código como a su trato en general supone tener que realizar una compra de éste (GIS comercial). Se pretende que la licencia del GIS seleccionado sea gratuita o de un coste no muy elevado. Se tendrá que llegar a un compromiso entre el precio y las características del GIS. Se valorará con un 15%.

Sistema Operativo: El sistema operativo en el cual trabaje el GIS es un aspecto a tener en cuenta pero no supone ningún tipo de obstáculo para seleccionar el software, ya que independientemente de ser en Linux o Windows se podrá trabajar con él sin ningún tipo de problema. Se valorará con un 5%.

La siguiente tabla representa los software con mejor resultado tras las primeras selecciones. Los primeros, de color naranja, son los que cumplen prácticamente todos los requisitos impuestos.

Los de color rosa son relativamente buenos, pero no acaban de encajar en el proyecto.

Por últimos los de color amarillo son descartados por no tener algún aspecto de importancia.

El proceso de investigación de GIS ha sido muy extenso debido a la gran cantidad de software que existen en el mercado. Se puede encontrar más información en el capítulo de Anexos.

	Mark	Program. 40%		Formatos 20%	BBDD 15%		Licencia 15%		Georef. & Orto. 5%		SO 5%	
ARCGIS	8,50	VB C++ C#	10	10	SI	10		0	SI	10	Cualquier Conocido	10
GvSIG	8,20	Java	8	8	SI	9	GNU GPL	9	Geo	6	Win Linux Mac OSX	8
QUANTUM	8,20	C++	8	8	PostGre SQL	9	Soft Libre	9	Geo	6	Windows Linux	8
TerraLib	7,50	C++	8	5	SI	9	Soft Libre	9	-	5	Windows Linux	7
GRASS	7,30	Código Abierto	8	7	SI	7	Soft Libre	7	Geo	6	Linux	6
ERDAS	7,25	C++	7	10	SI	10		0	SI	10	Win2000 WinXP	9
PostGIS	6,35	C C++ Java PLRuby	6	7	SI	6	GNU	7	?	5	?	7
Map Marker	6,10		5	8	SI	8	Version Pro 300€	4	Geo	7	Windows	7
GDAL / OGR	5,75	C++	7	8	NO	0	Soft Libre	6	NO	0	Windows	9
HidroGIS	5,75	Java	6	6	SI	5	GNU	7	?	2	?	5
Kosmos	5,50	Java	5	5	SI	7	GPL	6	Geo	5	Linux Windows	6
MiraMon	3,95		0	7	SI	7	Licencia 228€	5	SI	7	Windows	8

2.2.3. Conclusiones de Selección de GIS

Una vez analizados los sistemas GIS en profundidad se debe llegar a una conclusión sobre qué camino se debe seguir, y para ello se debe elegir uno de los software aquí mencionados.

Como aspecto positivo de esta etapa, se valora la variedad de posibilidades. Esto permite tener más alternativas a la hora de empezar a trabajar.
En consecuencia, el aspecto negativo, es el que brinda cualquier decisión: el saber si este es el camino correcto.

Tras muchas investigaciones, y analizar qué brinda cada programa, se ha llegado a las siguientes conclusiones.

Lo importante es tener una herramienta capaz de solventar de antemano los problemas básicos que genera el trabajo con mapas. De esta manera se deberá centrar casi toda la atención en desarrollar la nueva aplicación. También es importante tener las herramientas necesarias para poder solventar futuros problemas lo más rápido posible.

Dicho esto, se ha considerado que el pack de desarrollo de ArcGIS, SDK de ESRI, podrá adecuarse muy bien a los intereses de SKY-EYE, si bien es verdad que se debe asumir un gasto adicional. El valor de la licencia de desarrollo es de 1750€. Se debe también pagar 500€ por cada puesto de trabajo en el que se ejecute la aplicación resultante.

Esta herramienta requiere de licencia de desarrollo, pero ante la posibilidad de poder llegar a la meta de SKY-EYE supone una inversión muy necesaria y que garantiza un buen trabajo.

2.3. Plan de vuelo

La aplicación deberá cargar un plan de vuelo y visualizarlo correctamente. Este plan de vuelo ya estará diseñado. Por el momento la única función que se deberá desarrollar es la de guardar toda la información que contenga en una estructura de clases y visualizarla correctamente.

Para conseguir esto es necesario saber qué aspecto tiene el plan de vuelo. Es decir, en qué formato se encuentra, y qué estructura, y así crear un sistema de lectura del mismo.

Esta información está documentada en el “Flight Plan Specification” facilitado por Eduard Santamaría de ICARUS (EPSC) (ver Anexo H). En él se explica el modelo conceptual de los planes de vuelo. Aparece por primera vez en el proyecto el concepto de WayPoint, Fix, Leg, Stage, etc.

2.3.1. Modelo conceptual

- ***Plan de Vuelo***

Un Plan de vuelo especifica el patrón seguido por el avión. Todos los vuelos requieren un plan de vuelo principal y único, aunque pueden existir planes de vuelos alternativos. Los planes de vuelo tienen asociado un nombre y una descripción.

- ***Stage***

Cada plan de vuelo está formado por una secuencia de Stages (Etapas) que siguen una lógica. No se permitirá tener una etapa de aterrizaje inmediatamente después del despegue.

Los Stages están compuestos de Legs ordenados, y constituyen los bloques del plan de vuelo a alto nivel. Cada Stage tiene asociado un nombre y una descripción. Existe un número finito de descripciones, entre ellas el Despegue, Ruta, Misión, Ruta de Llegada, Aproximación, Aterrizaje, etc.
Los Stages tienen un Leg inicial y un leg final, que delimitan el inicio y final de dicho stage.

- ***Leg***

Un Leg especifica la ruta entre dos Waypoints. El parámetro *dest* especifica cual es el waypoint de destino. Todos los legs tienen un waypoint de inicio y uno de final.

Existen distintos tipos de legs. Entre ellos están el initial fix, que determina el comienzo del plan de vuelo, el track to fix representa una línea recta entre un waypoint y otro, el direct to fix representa una línea recta entre un punto del

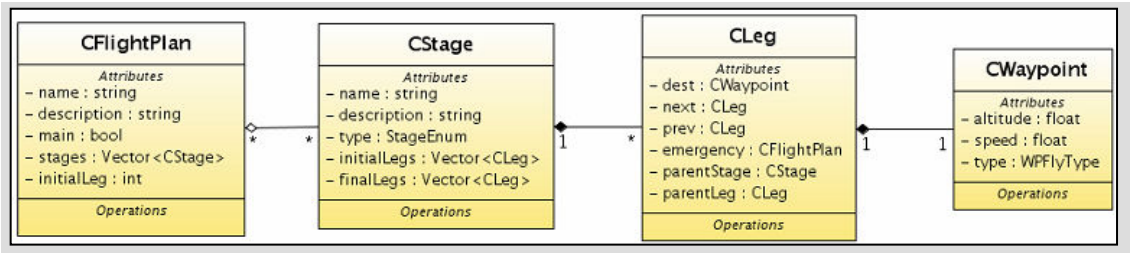
espacio y un waypoint, el radius to fix define la trayectoria entre dos waypoints describiendo una ruta en arco, con un radio y una dirección determinada, entre otros.

- ***Waypoint***

Un waypoint es una posición geográfica definida por una latitud y una longitud. Hay dos tipos de waypoint, los que tienen nombre y los que no. A los que tienen nombre se les llamará Fix. Estos contienen un nombre y una descripción. Un buen ejemplo de fix puede ser un Aeropuerto. Todos los waypoints tienen asociada una velocidad de paso y una altura.

La Figura 6, define gráficamente el modelo conceptual de un Plan de Vuelo.

Figura 6. Modelo conceptual Plan de vuelo



2.3.2. Material

ICARUS ha facilitado, además del documento de especificación, un XSD que define la estructura de plan de vuelo, y un plan de vuelo de prueba en formato XML para comenzar a trabajar.

2.4. Estación de Seguimiento

2.4.1. Introducción

La estación de seguimiento irá recibiendo paquetes de telemetría enviados por el UAV. Estos paquetes contienen toda la información referente al estado del UAV en el momento de envío, entre ella su posición.

La idea es muy similar a la de plan de vuelo. No se debe implementar nada nuevo. Se recibirán paquetes de telemetría y se deberá desarrollar un sistema capaz de almacenar esta información y visualizarla.

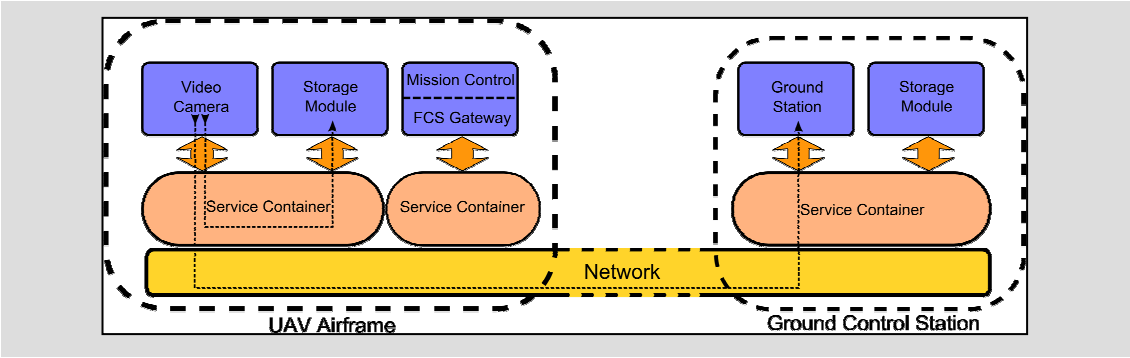
Para conseguir esto es necesario saber qué aspecto tiene un paquete de telemetría. Es decir, en qué formato se encuentra, y qué estructura, y así crear un sistema de lectura de ellos.

2.4.2. Información como un servicio

Toda la información relacionada con los datos que se recibirán una vez el avión haya despegado puede encontrarse en el documento "Service Abstraction Layer for UAV Flexible Application Development" facilitado por Pablo Royo de ICARUS (EPSC) (ver ANEXO I).

En este documento se explica el funcionamiento de envío de datos. Si se requiere de algún dato del UAV, se debe realizar una suscripción al servicio de envío. Existirán muchos usuarios de este servicio, que está conectado al UAV a través de una red de comunicaciones. La Figura 7 explica gráficamente el sistema:

Figura 7. Red de comunicaciones



2.4.3. Paquetes de telemetría

El formato aún no está definido. Lo que sí existe es la idea de acumulación de paquetes. Se podrá recibir paquetes de información específica, como por ejemplo la posición del UAV en un determinado momento. Pero un paquete de telemetría puede tener dentro otros paquetes como por ejemplo el de posición y el de ángulos.

Los distintos paquetes de información que podemos obtener están detallados al final del documento facilitado por Pablo Royo (ver ANEXO I). Entre ellos están los paquetes de posición, ángulo, aceleración, velocidad, tiempos, etc.

Al no estar establecido el formato, se deberá crear uno para poder trabajar con la aplicación.

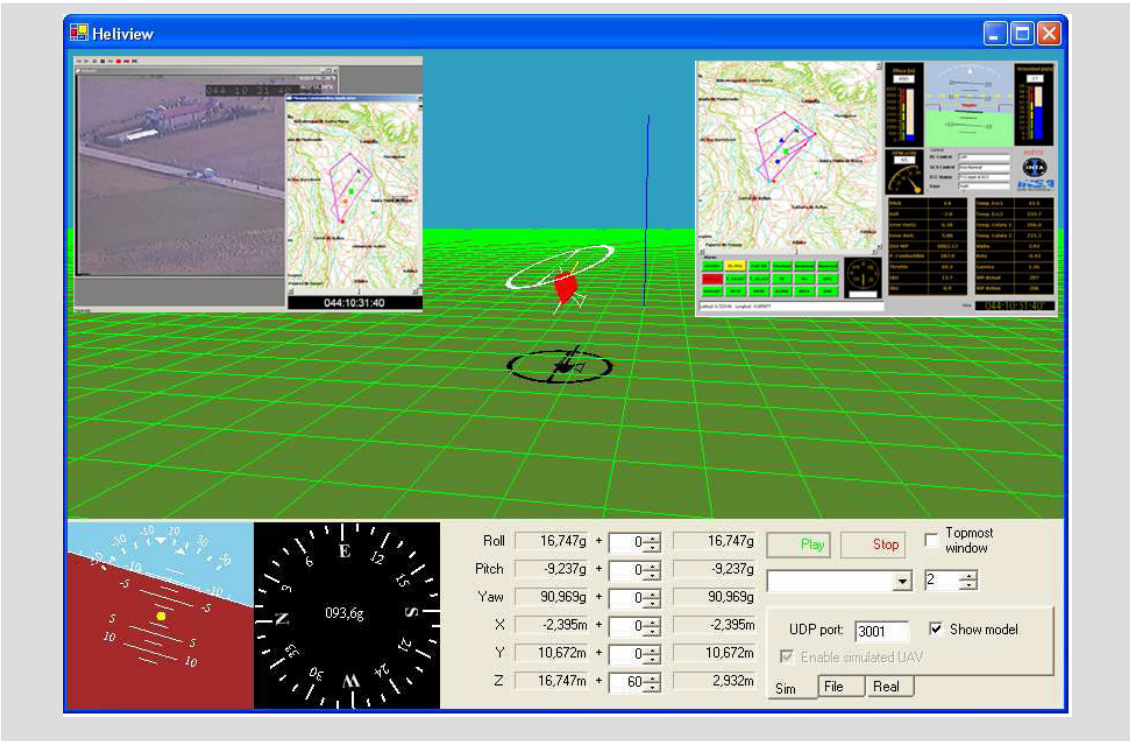
2.4.4. Paquetes de misión

Al haber muy poca información desarrollada respecto a este tema en el seno del grupo ICARUS, se han facilitado ideas para poder implementar una simulación en ésta aplicación. Existirán distintos tipos de paquete de misión, uno con la descripción de contenido (una fotografía por ejemplo), otro tipo con la fotografía en baja resolución dentro del paquete, y el último con la fotografía sin modificar. La diferencia de paquetes es el tamaño que ocupan, ya que no

siempre se dispondrá del ancho de banda necesario para transmitir. De esta manera, a pesar de no tener la información completa, se podrá tener algún tipo de descripción.

Se diseñarán tres tipos de paquetes con la información que se crea necesaria para poder hacer algún tipo de simulación. Al no tener un flujo de información desarrollado en el UAV, la tasa de recepción de paquetes será simulada por un *timer* interno de nuestra aplicación.

Figura 8. Captura de una Estación de seguimiento



2.5. Cartografías y ortofotos

Se debe disponer de una base para referenciar información. Esta base puede ser tanto fotografías como planos.

Ortofotos
Las ortofotos son fotrografías aéreas ortonormalizadas. Estas fotos tienen la peculiaridad de que parecen ser sacadas desde el infinito, es decir, todos los puntos proyectados en esa fotografía equidistan del foco.
Con esto se consigue que la foto cuadre perfectamente con un plano cartográfico ya que tienen el mismo *eje de coordenadas.*

Cartografías
Representa de forma geométrica lo que hay en la realidad. De esta manera no encontramos imágenes reales, sino una representación de ellas.

Tanto una ortofotos como cartografías tienen asociada una referencia espacial, y por ello cada punto del mapa, además de tener un contenido gráfico, tiene unas coordenadas.

Esta información será esencial para poder referenciar geográficamente lo que queramos representar. Se adquirirá información relacionada con la zona del Vallés Occidental, ya que ahí es donde se realizarán las primeras pruebas de vuelo de los UAV. Estas ortofotos y cartografías están disponibles en el Instituto Cartográfico de Cataluña.

CAPÍTULO 3. ESPECIFICACIÓN

3.1. Objetivos concretos del TFC

Llegado a este punto, y habiendo situado los objetivos generales, el entorno de trabajo, y estimando algunos de los posibles problemas a resolver, se debe concretar los pasos a seguir de forma más rigurosa.

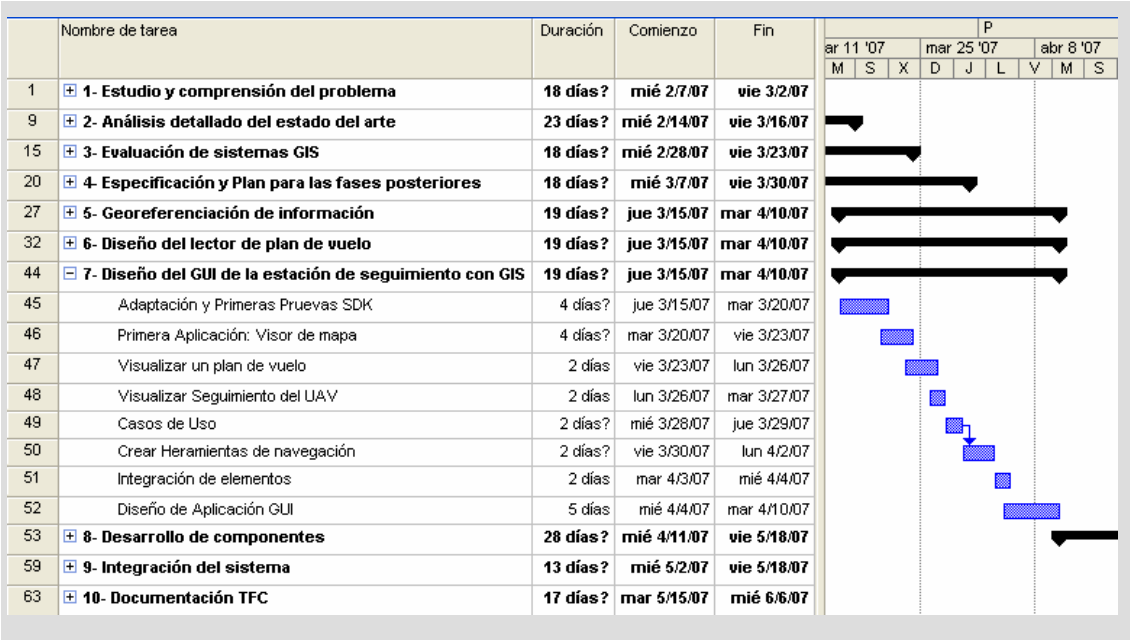
Para ello se comenzará enumerando los objetivos de este TFC:

- Adaptación y primeras pruebas con el SDK de ESRI. Se intentará adaptarse al trabajo con un SDK y en particular al de ESRI.
- Primera aplicación: Visor de mapa y controles básicos. Programar la primera aplicación relacionada con el proyecto.
- Casos de uso. Estudiar el tema de casos de uso y aplicarlo a la aplicación.
- Visualizar plan de vuelo. Dado un plan de vuelo, o habiendo creado uno, visualizarlo correctamente en un mapa.
- Visualizar seguimiento del UAV. Simular el envío de paquetes y pintarlos sobre un mapa.
- Crear herramientas de navegación. Crear todas las herramientas necesarias para una correcta navegación y exploración del mapa y sus elementos.
- Integración de todos los puntos. Integrar todos los puntos propios de este TFC.
- Integración de las otras partes del proyecto. Agrupar el trabajo desarrollado por los tres miembros del grupo.
- Diseño final de la aplicación

3.2. Planificación detallada

La planificación temporal de las etapas de este TFC en particular queda definida tal y como muestra la Figura 9:

Figura 9. Tareas Individuales



	Nombre de tarea	Duración	Comienzo	Fin
1	**1- Estudio y comprensión del problema**	**18 días?**	**mié 2/7/07**	**vie 3/2/07**
9	**2- Análisis detallado del estado del arte**	**23 días?**	**mié 2/14/07**	**vie 3/16/07**
15	**3- Evaluación de sistemas GIS**	**18 días?**	**mié 2/28/07**	**vie 3/23/07**
20	**4- Especificación y Plan para las fases posteriores**	**18 días?**	**mié 3/7/07**	**vie 3/30/07**
27	**5- Georeferenciación de información**	**19 días?**	**jue 3/15/07**	**mar 4/10/07**
32	**6- Diseño del lector de plan de vuelo**	**19 días?**	**jue 3/15/07**	**mar 4/10/07**
44	**7- Diseño del GUI de la estación de seguimiento con GIS**	**19 días?**	**jue 3/15/07**	**mar 4/10/07**
45	Adaptación y Primeras Pruevas SDK	4 días?	jue 3/15/07	mar 3/20/07
46	Primera Aplicación: Visor de mapa	4 días?	mar 3/20/07	vie 3/23/07
47	Visualizar un plan de vuelo	2 días	vie 3/23/07	lun 3/26/07
48	Visualizar Seguimiento del UAV	2 días	lun 3/26/07	mar 3/27/07
49	Casos de Uso	2 días?	mié 3/28/07	jue 3/29/07
50	Crear Heramientas de navegación	2 días?	vie 3/30/07	lun 4/2/07
51	Integración de elementos	2 días	mar 4/3/07	mié 4/4/07
52	Diseño de Aplicación GUI	5 días	mié 4/4/07	mar 4/10/07
53	**8- Desarrollo de componentes**	**28 días?**	**mié 4/11/07**	**vie 5/18/07**
59	**9- Integración del sistema**	**13 días?**	**mié 5/2/07**	**vie 5/18/07**
63	**10- Documentación TFC**	**17 días?**	**mar 5/15/07**	**mié 6/6/07**

Como se puede apreciar, éste diagrama no coincide con el propuesto al comienzo de éste documento. Las irregularidades y problemas que se ha sufrido para causar este hecho se explicarán en el capitulo de Balance.

Los tres proyectistas dispondrán de 19 días para conseguir los objetivos propios. En el caso particular de este proyecto se tiene en cuenta la curva de aprendizaje de la programación con el SDK de ESRI. Las primeras tareas tienen una mayor carga ya que son la puesta en contacto con las herramientas de ArcGIS.

Una vez se tenga el esqueleto de la aplicación, es decir, un visor de cartografía, se podrá comenzar a pintar información útil.

Se disponen de cuatro días para pintar el plan de vuelo y el seguimiento. Posiblemente se pueda aprovechar mucho código para ambas funciones.

Las tareas finales son las de desarrollar herramientas para que el usuario navegue de forma mas cómoda por la aplicación. Desde el estudio de Casos de Uso, hasta la integración de elementos, se intentará desarrollar dichas herramientas.

Finalmente se diseñará un prototipo de interfaz de usuario para la aplicación. Será la versión definitiva de estos objetivos individuales, para que una vez se junten todas las partes, el grupo en conjunto realice los cambios oportunos.

3.3. Funcionalidad y Casos de uso

3.3.1. Funcionalidad

El objetivo de la aplicación es que un usuario pueda visualizar toda la información relacionada con el vuelo de uno o varios UAV.

Para ello la primera función a la que se recurre es la de cargar un mapa o cartografía. Sobre él se enseñará la información de que se dispone inicialmente: el plan de vuelo.

Se deberá poder seleccionar lo que se desea visualizar. Por ello debe haber herramientas básicas para ver sólo waypoints, sólo legs, de todo el plan de vuelo o parte de él.

Mientras se están recibiendo los paquetes de telemetría el usuario puede querer verlos al momento o no. Para ello se deberán crear herramientas de visualización de plan de vuelo o telemetría independiente una de la otra.

El mapa tendrá sus propias herramientas de navegación para poder aumentar el zoom, disminuirlo, moverse hacia los lados, o centrar el mapa en la posición inicial.

El usuario tiene que tener acceso a toda la información que el plan de vuelo o telemetría permita. Esta información podrá ser accedida a través de los elementos visuales (pintados sobre el mapa), o a través de una lista cargada con todos los elementos. Cuando accedamos a la información a través de un elemento visual, o a través de una lista, el elemento gráfico deberá cambiar de color para representar que está siendo seleccionado, y se seleccionará el elemento de la lista con el mismo fin.

Se deberán implementar herramientas de diseño con el fin de que el usuario pueda cambiar colores o estilos de visualización.

Finalmente, se ha decidido que el usuario no deberá elegir por sí mismo las ortofotos, sino que al cargar el plan de vuelo deseado, el sistema busca las ortofotos en las que dicho plan de vuelo puede ser pintado. De esta manera el usuario no necesita corresponder el plan de vuelo con unas ortofotos, ya que se hará automáticamente. Por ello el usuario debe tener un editor de fotos, que acceda a una base de datos, para cargar o eliminar fotografías de ella.

3.3.2. Casos de uso

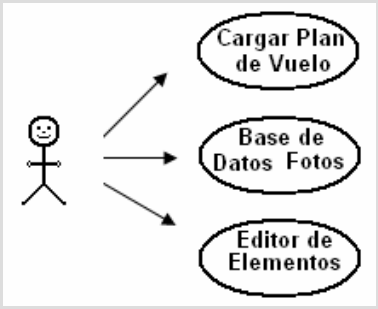
El diagrama de casos de uso representa la forma en cómo un Usuario(Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en cómo los elementos interactúan (operaciones o casos de uso).

Para empezar se debe identificar los actores que interactúan con el sistema. Al ser un visor de información, por ahora no habrá más que un operario que busque dicho fin.

Figura 10. Casos de Uso 1

Ahora ver qué puede hacer este actor con la aplicación.

Figura 11. Casos de Uso 2

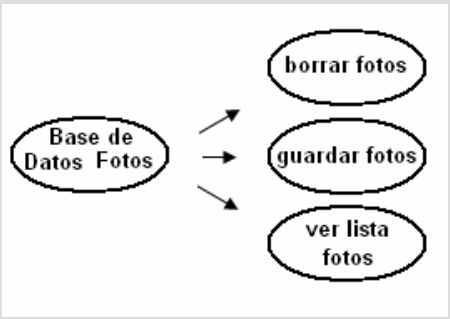


Algunos usos dependen de otros:

Figura 12. Casos de Uso 3

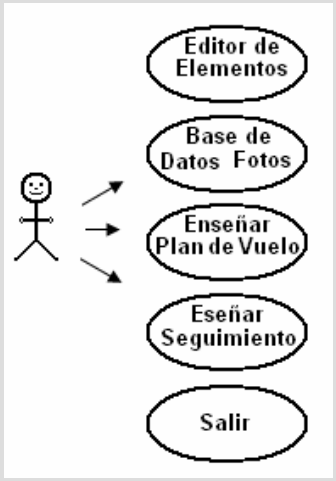
El actor puede interactuar de varias maneras con la base de datos:

Figura 13. Casos de Uso 4



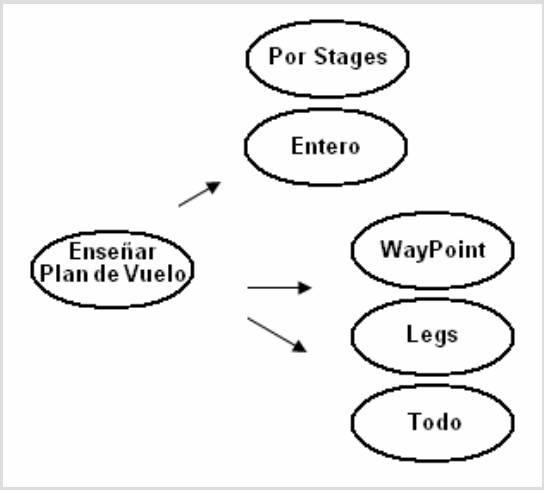
Una vez esté cargado el mapa, aparecen nuevos casos de uso:

Figura 14. Casos de Uso 5



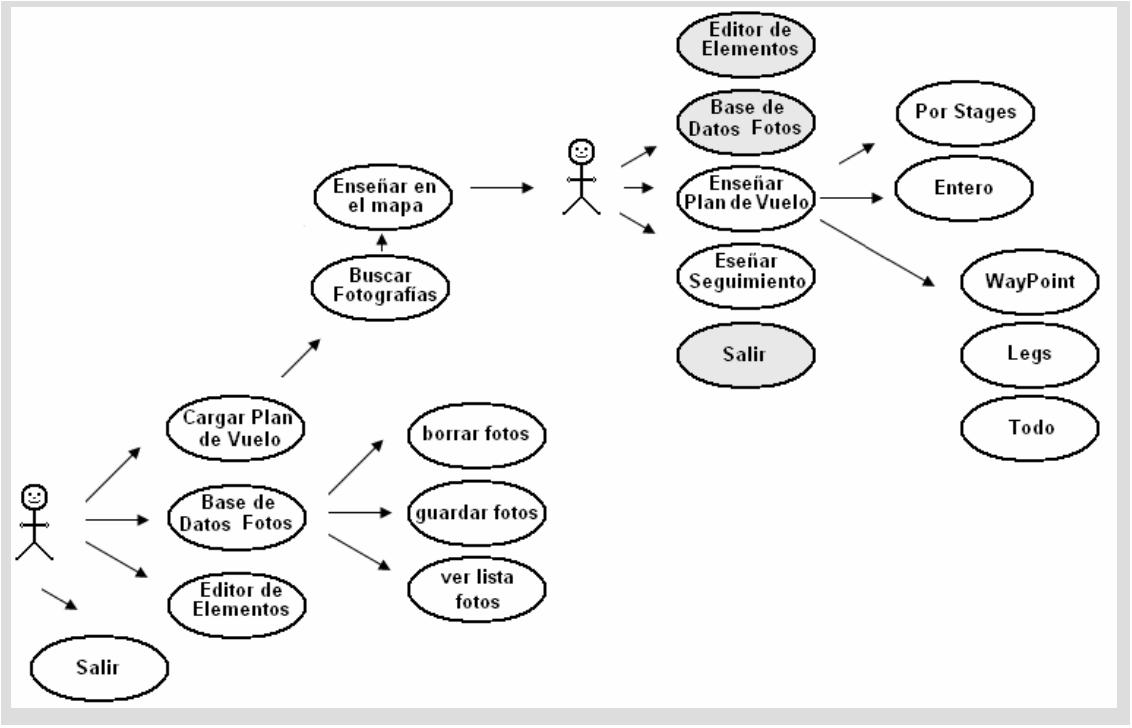
Y alguno de ellos tiene otros usos asociados:

Figura 15. Casos de Uso 6



Y de los datos que estén siendo mostrados, se podrá ver su información asociada. El diagrama completo de casos de uso es el siguiente:

Figura 16. Casos de Uso 7



3.3.3. Tablas de Casos de Uso

En este apartado se enseñarán las tres principales funciones de que se dispone al iniciar el visor de plan de vuelo.

Cargar Plan de Vuelo:

Caso de Uso: Cargar Plan Vuelo
Actores: Gestor (iniciador)
Propósito: Cargar un plan de Vuelo
Resumen: --
Tipo: Primario Esencial

Curso típico de acciones:

Acciones del actor	Respuesta del sistema
1. Un Usuario Gestor comunica al sistema que quiere cargar un plan de vuelo	**2.** El sistema busca el plan de vuelo y retorna las fotos que visualizan dicho plan.
3. El usuario elige el grupo de fotografías	**4.** El sistema enseña las fotos en el panel central

Cursos alternativos:

No cargar el plan y entrar al gestor de fotografías.

Gestor de Fotografías: Crear Grupo de Fotos:

Caso de Uso: Gestor de Fotos
Actores: Gestor (iniciador)
Propósito: Abrir el Gestor de Fotos y guardar grupo nuevo
Resumen: --
Tipo: Primario Esencial

Curso típico de acciones:

Acciones del actor	Respuesta del sistema
1. Un Usuario Gestor comunica al sistema que quiere abrir el gestor	**2.** El sistema abre el gestor
3. El usuario comunica que quiere crear nuevo grupo	**4.**El sistema activa las herramientas de creación: buscar path, guardar, etc.
5. El usuario selecciona el path donde se encuentran las fotos.	**6.** El sistema enseña un listado de todas las fotos dentro de ese path.
7. El usuario rellena los campos y pide guardar.	**8.** El sistema crea un archivo XML con toda la información ya existente, y la nueva.

Cursos alternativos:

Visualizar los grupos o borrarlos.

Editar elementos gráficos

Caso de Uso: Configuración Elementos Gráficos
Actores: Gestor (iniciador)
Propósito: Cambiar configuración de elementos gráficos
Resumen: --
Tipo: Secundario No Esencial

Curso típico de acciones:

Acciones del actor	Respuesta del sistema
1. Un Usuario Gestor comunica al sistema que quiere abrir gestor de configuración visual	**2.** El sistema abre el gestor. La información que visualiza la busca en la estructura de clases (que han sido cargadas de un archivo XML inicial).
3. El usuario comunica qué cambios desea hacer.	**4.** El sistema guarda los cambios en el archivo XML y actualiza la estructura de clases con la nueva información.

Cursos alternativos:

--

CAPÍTULO 4. DISEÑO

4.1. Plataforma de desarrollo

4.1.1. .NET

El .NET Framework es un entorno multi-lenguaje para la construcción y ejecución aplicaciones de escritorio, aplicaciones web, de dispositivos móviles, etc.

Common Language Runtime (CLR)
EL CLR juega un importante papel tanto para la ejecución de los componentes de las aplicaciones como de su desarrollo. Durante la ejecución, el CLR es responsable de supervisar la gestión de memoria, lanzar e interrumpir hebras de ejecución y procesos, asegurar las políticas de seguridad, así como satisfacer las dependencias que en tiempo de ejecución pueda tener un componente. Todo lo anterior hace que durante el desarrollo sea mucho más sencillo.

Librería unificada de clases
EL .NET framework proporciona a los desarrolladores un conjunto unificado de librerías (APIs) de clases jerárquicas, orientadas a objetos y extensibles. El conjunto de APIs es común para todos los lenguajes de programación, lo que permite herencia, gestión de errores y depuración a través de múltiples lenguajes de forma transparente. Todos los lenguajes acceden al framework de forma similar por lo que el programador puede elegir el lenguaje que prefiera sin perjuicio para las funcionalidades de la aplicación que esté desarrollando.

4.1.2. Visual Studio

Visual Studio .NET es un completo entorno integrado de desarrollo para la construcción de aplicaciones Web ASP, servicios Web XML, aplicaciones de escritorio y aplicaciones móviles.
Todos los lenguajes de programación soportados – Visual Basic .NET, Visual C++ .NET, Visual C# .NET y Visual J# .NET (aunque pueden añadirse otros lenguajes) – usan el mismo entorno integrado de desarrollo (IDE), lo que permite compartir herramientas y desarrollar proyectos multi-lenguaje con mucha facilidad. Además, dichos lenguajes explotan al máximo las funcionalidades del .NET Framework, lo que proporciona acceso a un conjunto de tecnologías que simplifican considerablemente el desarrollo de aplicaciones.

4.1.3. XML

El lenguaje XML, entre otras cosas, proporciona un potente mecanismo para la descripción de datos estructurados. XML es un subconjunto de SGML optimizado para ser distribuido a través de la Web. El World Wide Web Consortium (W3C) define estándares XML para que dichos datos estructurados sean uniformes e independientes de la plataforma. Visual Studio .NET incluye una herramienta llamada XML Designer que permite editar ficheros XML y crear schemas XML.

4.2. ESRI Develper Kit

4.2.1. Introducción

ArcGIS es el punto de mira de todas las aplicaciones GIS del momento. Es el producto que mejores herramientas ofrece, y por ello también es de los más respetados a nivel mundial.

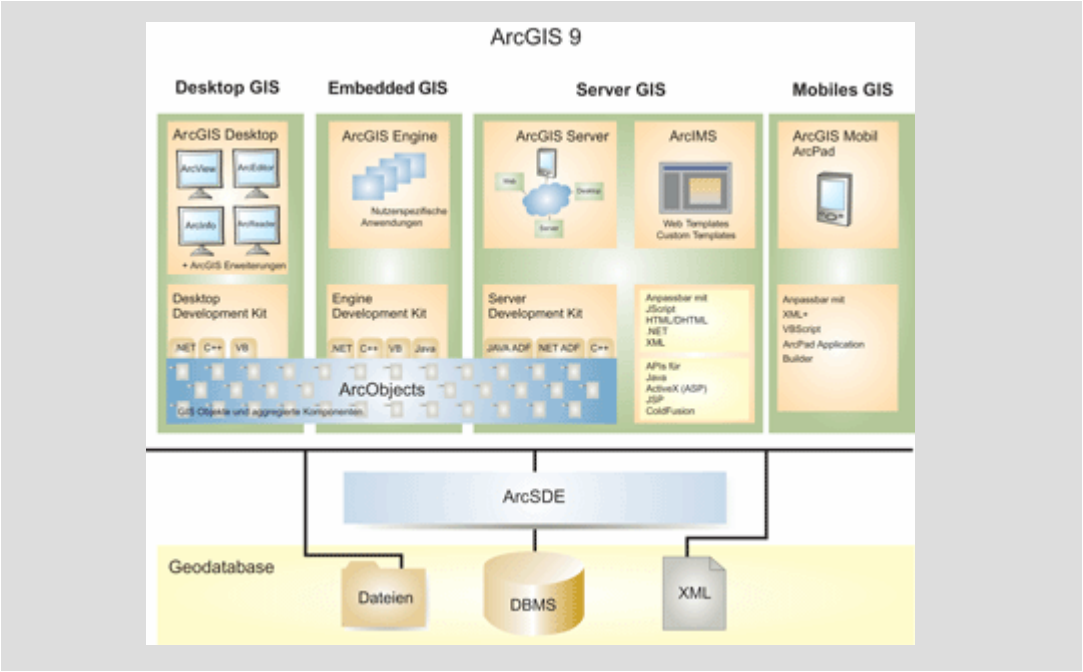
Ofrece, a través del kit de desarrolladores, una serie de herramientas para poder programar nuevos programas relacionados con el mundo de la geomática. De esta manera, el programador consigue tener a su disposición mucho código de funciones bien resueltas de manera rápida para estructurarlas a su conveniencia.

Se puede acceder a todos las funciones de escritorio que se aprecian en los programas de ArcGIS, ya sea de dibujar una línea, como cargar un mapa a través de WMS, etc.

Este development kit provee acceso a una larga colección de componentes *ArcObjects* (los componentes con los que se construye ArcGIS) y también incluye varios controles de desarrollo con el fin de crear software de gran calidad. Los componentes visuales están disponibles como controles .NET, controles JavaBeans, y ActiveX. El development kit puede ser utilizado tanto en Windows, Linux, o el sistema operativos Solaris.
Como ayudas adicionales a los programadores, el kit viene acompañado de un Development kit help, en el que se describen las interficies, funciones, métodos y todo lo que se pueda encontrar en el development kit y genere algún tipo de duda, así como ejemplos de programas. También se puede consultar el foro de programadores que ofrece ArcGIS.

4.2.2. Arquitectura de ArcGIS

Figura 17. Diagrama de Arquitectura de ArcGIS



En el diagrama de la Figura 17 se puede apreciar de forma esquemática las distintas plataformas de ArcGIS, y cómo se relacionan entre sí. Como se puede apreciar a la izquierda de la Figura 1, Desktop Development Kit trabaja sobre ArcGIS Desktop, Engine sobre ArcGIS Engine, y Server sobre ArcGIS server. Se crean los ArcObjects (ANEXO G: en librería Controls), y el último paso es decir qué se pretende hacer con esta información.

4.2.3. Librerías con más importancia para SKYEYE

Como es de esperar no todas las librerías del developer kit son necesarias. Se valora la gran variedad de posibilidades que permite el Developer kit, pero para empezar se utilizarán muy pocas. Entre ellas System, SystemUI, Geometry, Display, Carto, GeoProcessing y Controls (ver ANEXO G). Cuando se acaben las primeras etapas del proyecto se volcará bastante importancia a las que trabajen con bases de datos y servicios Web.

4.2.4. ArcGIS Toolbar

Corriendo el Developer Kit sobre la plataforma .NET, en ella se añaden diversos controles visuales: ArcGis Windows Forms. Estos controles son los siguientes:

MapControl: Este control corresponde al ‘data view’ de la aplicación desktop ArcMap y encapsula el objeto Map. Con este control podemos cargar todo tipo de mapas, también se puede usar para crear mapas.
PageLayoutcontrol: Este control corresponde al ‘layout view’ de la aplicación desktop ArcMap y encapsula el objeto PageLayout.

TOCControl (Table of contents control): Este control debe trabajar conjuntamente con otro control como: Mapcontrol, PageLayoutcontrol, Globecontrol o SceneControl. TOCControl se encarga de visualizar los mapas, capas o simbologías contenidos en el control con el que está trabajando.

ToolbarControl: Este control debe trabajar conjuntamente con otro control como: Mapcontrol, PageLayoutcontrol, Globecontrol o SceneControl. El ToolbarControl tiene todo un panel de comandos, herramientas, menús, gamas de colores que trabajan sobre el otro control.

SceneControl: Este control corresponde al ‘3D View’ de la aplicación desktop ArcScene. Provee una manera de ver e investigar pequeñas capas de datos en tres dimensiones. Este control requiere la extensión 3D Analyst.

GlobeControl: Este control corresponde al ‘3D View’ de la aplicación desktop ArcGlobe. Provee una visión 3D de datos sobre una superficie de globo, trabajando sobre localizaciones geográficas verdaderas. Este control requiere la extensión 3D Analyst.

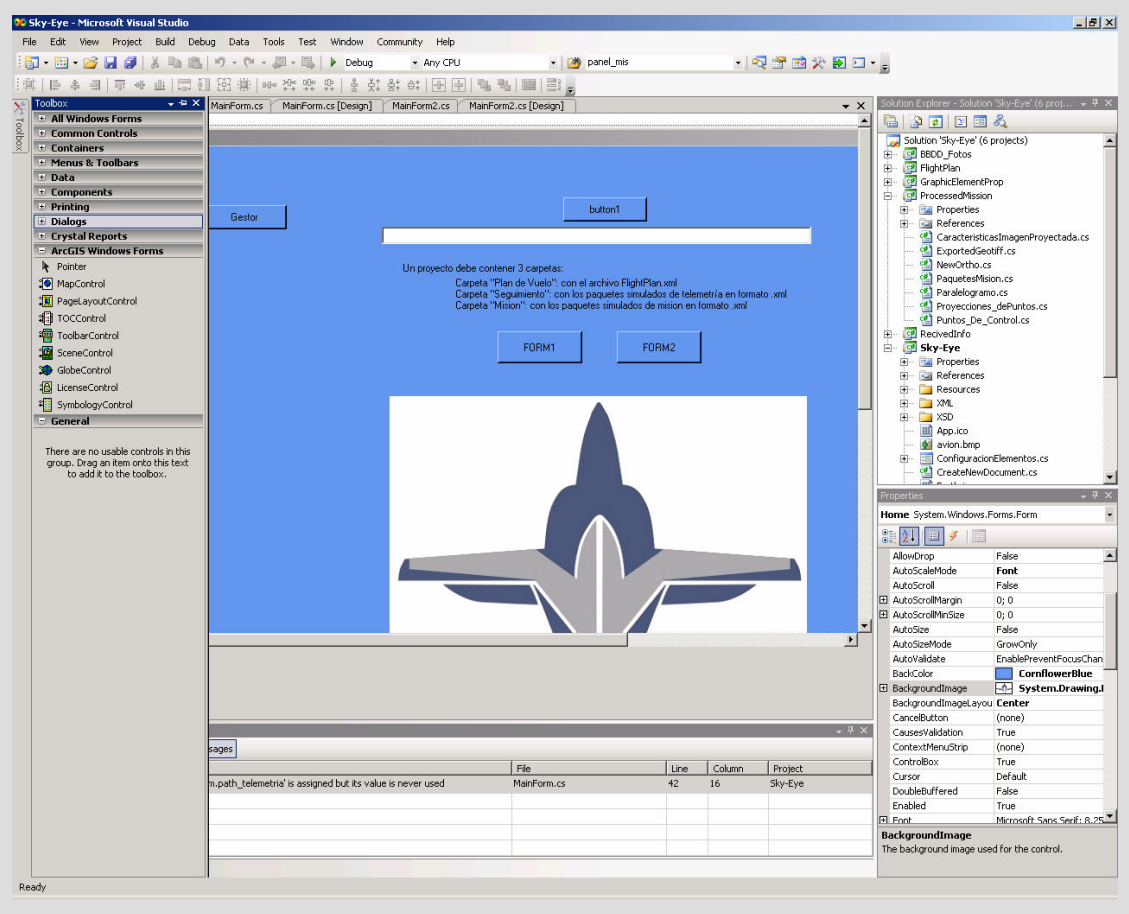
LicenseControl: Este control se utiliza para inicializar las herramientas con una licencia conveniente para que pueda funcionar con éxito sobre cualquier máquina en la que se despliegue la aplicación desarrollada.

Symbologycontrol: Este control se utiliza para visualizar el contenido de los archivos ‘*.ServerStyle’ y de simbologías personalizadas. El SymbologyControl permite a un usuario final seleccionar un símbolo, esto también se puede aplicar a una parte de la aplicación.

De esta forma se trabajará con herramientas GIS de la misma manera que se puede trabajar con botones, labels, paneles, etc, en la construccion de una aplicación .NET convencional.

En Figura 18 se puede ver las herramientas de ESRI dentro del Toolbox de Visual Studio.

Figura 18. Derecha de la imagen: ToolBar C# con SDK ESRI



4.2.5. Conclusiones

Una vez visto el Developer Kit, y haber investigado un poco su estructura, se puede apreciar mejor la grandeza de este conjunto de aplicaciones. Se reafirma la confianza en este software para conseguir los fines propuestos al principio del proyecto.

4.3. Estructura de clases

Antes de comenzar a trabajar con la representación gráfica de los elementos de un plan de vuelo, se deberá guardar toda la información que se obtenga en una estructura de clases. Los diagramas o estructuras de clases se encargan de crear el diseño conceptual de la información que utilizará el sistema, los componentes que se encargan del funcionamiento y la relación entre ellos.

Así mismo existirán distintos grupos de clases, que desempeñen tareas distintas. En concreto cuatro. Están las clases relacionadas con el plan de vuelo, las que tengan que ver con las fotografías disponibles, las que contengan la configuración visual de los elementos, y las que tengan que ver con los paquetes de telemetría o misión.

4.3.1. Plan de vuelo

La estructura de clases creada para trabajar con el plan de vuelo tiene un esquema muy similar al XSD facilitado por ICARUS. De esta manera se conserva la arquitectura propuesta.

4.3.2. Fotografías

Existirá una base de datos con una lista de fotografías. Esta lista contendrá el nombre de cada foto y las coordenadas del espacio que representan. Estarán agrupadas, y este grupo tendrá además unos campos descriptivos como el directorio, el formato, o la resolución.

Se tiene que tener presente la estructura para poder crear la estructura de clases. El mecanismo de creación es muy similar al del plan de vuelo.

4.3.3. Configuración visual de elementos

Esta clase guarda la información relacionada con la presentación de los elementos.
Cada línea o punto que represente una estructura de datos como un leg o un waypoint podrá ser configurada a gusto del usuario.
Una vez se haya hecho dicha configuración a través de un asistente de elementos, ésta se carga en la estructura de clases creada y deberá guardarse en un archivo XML, para que al cerrar la aplicación no se pierda dicha configuración.

Más información sobre la estructura de clases de este proyecto se puede encontrar en el TFC realizado por Borja López.

4.3.4. Telemetría y misión

Al margen de los datos que se obtengan al inicio de la aplicación, como son el plan de vuelo, las fotografías o la configuración de elementos, existirán datos que se recibirán en el transcurso de la aplicación. Estos datos son enviados por el avión, y pueden ser tanto de telemetría como el resultado de la ejecución de la misión.

Se debe crear otra estructura de clases que guarde esta información para poder ser visualizada.

4.4. Herramientas Gráficas

4.4.1. Capas y Container

Es interesante que la información pintada se pueda visualizar independiente de las cartografías. En los GIS existe el concepto de “Capas”. Estas capas se superponen al mapa, y se pueden pintar sobre ellas. De esta manera, al ocultar o visualizar una capa, se verán o no todos los elementos que ella contenga.

Existen muchos tipos de capas, en función de lo que se quiera pintar. Para pintar elementos se utilizar la GraphicLayer.

Tras mucha investigación en el tema, no se ha conseguido visualizar la información mediante este método. Aún queda por resolver si el problema es al pintar la capa, o al activar la capa en un mapa.

Por este motivo se ha buscado alternativas. La que mejor se adecua al problema es utilizar el contenedor del mapa. Un mapa posee un contenedor en el que se puede volcar elementos gráficos. Estos elementos se visualizarán al refrescar el mapa. En otras aplicaciones se introducen al contenedor elementos con información de texto (labels) para dar información al mapa.

Para poder elegir lo que se pretende mirar en un mapa, se deberá crear un criterio de selección de información del container. Se irá borrando y agregando elementos en función de lo que se quiera ver en ese momento. En el siguiente capítulo se explicará cómo la tabla de hash ha ayudado a solucionar este problema.

4.4.2. Pintar Elementos

Para pintar elementos se utilizarán las herramientas gráficas de ESRI y no las del Framework de .NET. El principal motivo es que si luego se quiere volcar esta información en un contenedor de ESRI, esta información deberá ser algún objeto de ESRI.

De esta manera, por ejemplo, no se utilizará el objeto Color del Framework de .NET, sino la interface IColor de ESRI.

Se crearán funciones específicas para pintar puntos, líneas rectas, o líneas curvas para estructurar el problema. Una vez se hayan creado estas funciones existirán otras más grandes que incluyan a las específicas. Por lo tanto un plan de vuelo, que como hemos explicado es un conjunto de waypoints y legs, será una función que llame a otra encargada de dibujar un punto cuando se tenga un waypoint, o a la que se encargue de dibujar una linea cuando se tenga un leg.

En el capítulo de implementación se desarrollarán estos temas a fondo.

4.5. Aspecto Visual de la aplicación

Es importante pensar en cómo va a ser la distribución de las herramientas de navegación y visualización sobre la pantalla. En este apartado se describe el diseño de las tres pantallas principales.

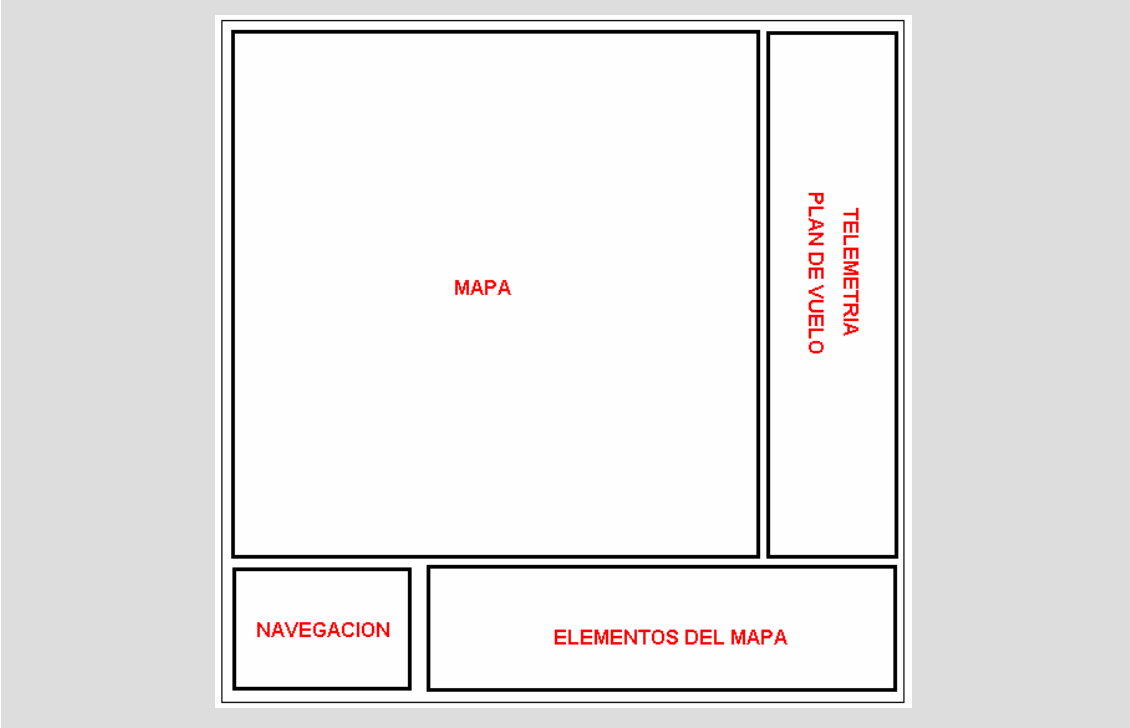
Como toda aplicación de visualización se deberá disponer de un amplio panel en el que se represente el mapa.

A la izquierda se dispondrá de un área para mostrar la información de plan de vuelo y telemetría. El usuario podrá interactuar con este panel para seleccionar lo que desea pintar.

Por debajo del mapa se tienen las herramientas de navegación y las herramientas de selección de elementos. Se podrán seleccionar los elementos desde estas herramientas.

El aspecto visual quedará distribuido de la siguiente manera:

Figura 19. Centro de la imagen: Gestor para configuración de Elementos



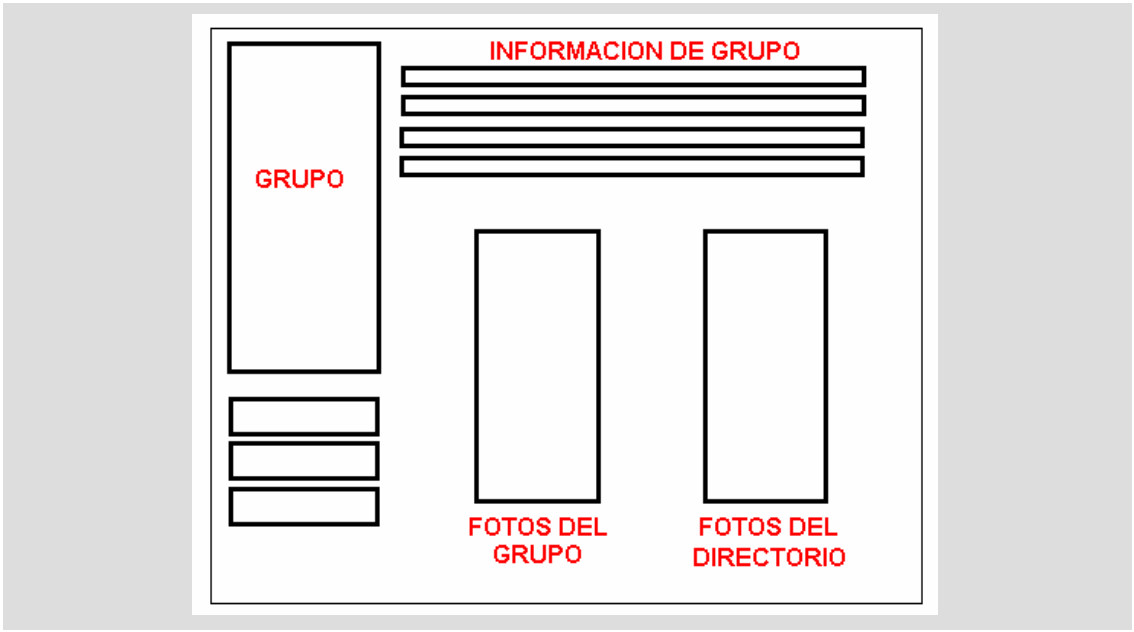
El gestor de fotografías deberá visualizar los grupos de fotos cargados, enseñar, borrar o crear nuevos.

A la izquierda de la pantalla se podrá ver en forma de listas los grupos que se tenga, y a la derecha se muestra la descripción asociada al grupo seleccionado.

Se dispondrá de botones para seleccionar si se desea guardar, ver o borrar.

El aspecto visual queda descrito por la siguiente figura:

Figura 20. Centro de la imagen: Gestor para configuración de Elementos

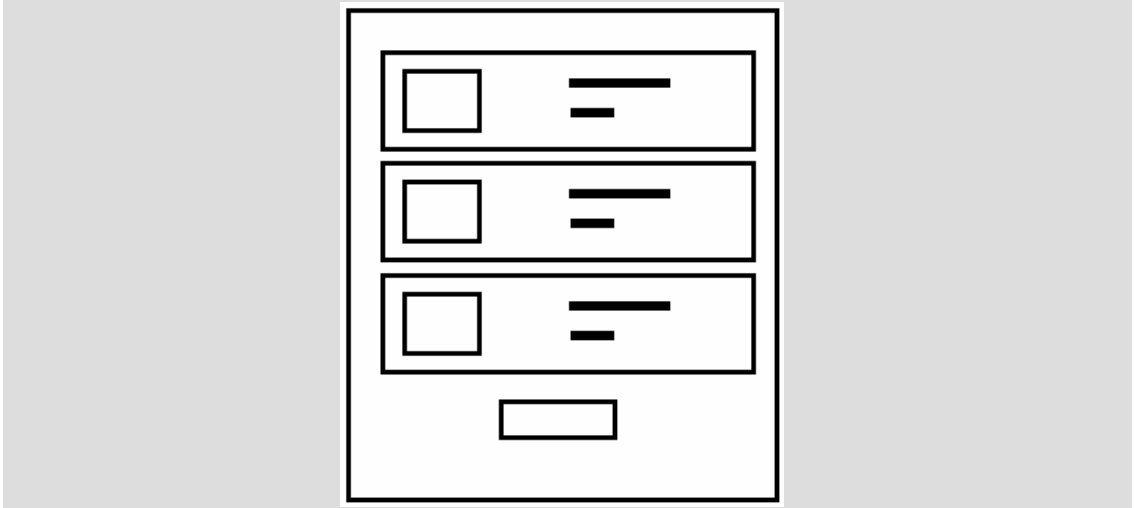


Los dos rectángulos de debajo enseñan las fotos que hay en la lista seleccionada, y a la derecha las fotos que existe en el path inicial del grupo.
De esta forma el usuario sabe si hay fotos del directorio que no se encuentran en el grupo.

Con respecto al gestor de colores y estilos de los elementos, el diseño será muy simple. Se ensañará el elemento, con su descripción al lado. El elemento se pintará de la misma forma que se hace en esta pantalla. Por ese motivo la elección del usuario se reproducirá en el mapa de igual manera.

Debajo un botón para guardar y salir.

Figura 21. Centro de la imagen: Gestor para configuración de Elementos



CAPÍTULO 5. IMPLEMENTACIÓN Y PRUEBAS

5.1. Implementación

Para comenzar a pintar, y teniendo en cuenta lo que se ha explicado anteriormente, se desarrollaran las funciones específicas de pintar un punto, una línea recta, un arco, y un ítem.

5.1.1. Pintar un punto: `DibPunto`

Esta función pide las coordenadas y un texto. Con las coordenadas se ubicará el punto, y el texto sirve para describir qué tipo de elemento es. De esta manera se puede distinguir un waypoint y un paquete de telemetría recibido. Los dos serán puntos pero gracias al texto que los identifica se pintarán de una forma u otra.

La función busca en la estructura de clases el color y estilo definido para el elemento en función del texto introducido.

Una vez esté creado el punto, la función no lo pinta, sino que devuelve un IPoint dentro de un tipo IElement. Esta interface es un tipo general, que puede tener dentro tanto puntos, como líneas o ítems.

En resumen a la función se le pasa unas coordenadas, un texto, y devuelve un IElement listo para ser agregado al container.

5.1.2. Pintar una recta: `DibLinea`

Funciona de forma similar que el punto. La principal diferencia es que se le debe pasar dos pares de coordenadas. La función devuelve un IPolyLine dentro de un IElement.

De la misma manera que con el punto, se le debe pasar un texto para que la función busque en la estructura de clases el color y estilo de la linea.

5.1.3. Pintar un arco: `DibArco`

Prácticamente igual que `DibLinea`. La única diferencia es que se le debe pasar un radio y una dirección de giro.

5.1.4. Pintar un ítem: `DibItem`

Esta función servirá para pintar imágenes prediseñadas de poco tamaño, como por ejemplo un avión. Se le pasarán las coordenadas y un texto para saber qué ítem dibujar.

5.1.5. Pintar un plan de vuelo

La forma de pintar un plan de vuelo tiene mucho que ver con su modelo conceptual.

El principal problema que se encuentra es el siguiente:

Cada leg tiene un campo destino, que puede estar rellenado por un fix o por unas coordenadas. Un leg también tiene un campo next y otro prev, que se refieren a los puntos de donde vienen y a donde van. Estos puntos no son coordenadas sinó identificadores, es decir, texto. Esto desencadena una serie de problemas:

- No se puede pintar un leg independiente del plan de vuelo sin realizar más búsquedas, ya que no se tienen las coordenadas de inicio y final, sino unos identificadores. Por lo tanto antes de pintar un leg aislado, se debe preguntar a qué coordenadas corresponde cada identificador.
- El destino de un leg puede estar definido por un Fix o por un par de coordenadas. En caso de tener coordenadas, saber a dónde se va es sencillo, pero si se tiene un Fix, se debe recorrer la lista de fixes para buscar el que interesa y preguntar cuáles son sus coordenadas.

Esto con lo que refiere a los legs. Para pintar waypoints o fixes se debe recorrer tanto la lista de fixes como el plan de vuelo, buscando cada uno para ser pintado.

Primero se pintarán los legs, y luego los waypoints con el fin de poder ver los puntos sobre las líneas y no al revés. Además se ha creído interesante pintar el plan de vuelo por Stages, con el fin de poder visualizar un stage independiente de otro.

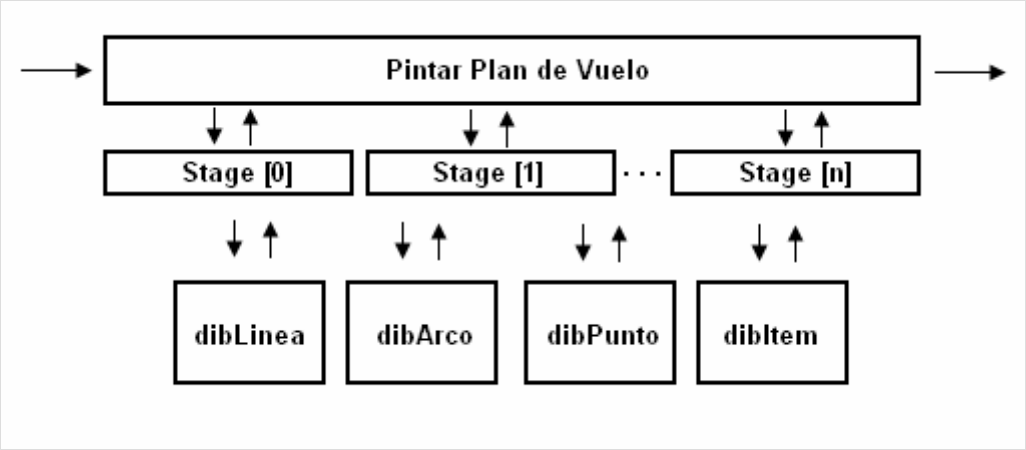
Se puede apreciar que el sistema de pintado no es sencillo. Se debe pintar el plan de vuelo tal y como se lee, teniendo en cuenta que al leer se generan muchas operaciones tanto de memoria como de búsqueda.

Por lo tanto el plan de vuelo se pintará de la siguiente manera:

- La función pintar plan de vuelo llamará a la función pintar un stage tantas veces como stages tenga.
- Esta función pintará todos los legs de su stage, y luego todos los waypoints y fixes que tenga.

- Cada línea será pintada por `DibLinea`, y cada punto será pintado por `DibPunto`.
- Se irán cargando todos los elementos al container para que una vez hecho el refresh (refrezco del mapa) aparezcan pintados.

Figura 22. Diagrama de la función de pintado de un plan de vuelo



De esta manera se ha conseguido pintar todo el plan de vuelo. Ahora será interesante plantearse la idea de rescatar la información de forma visual. Es decir, que al pasar el ratón por algún elemento se consiga ver su información.

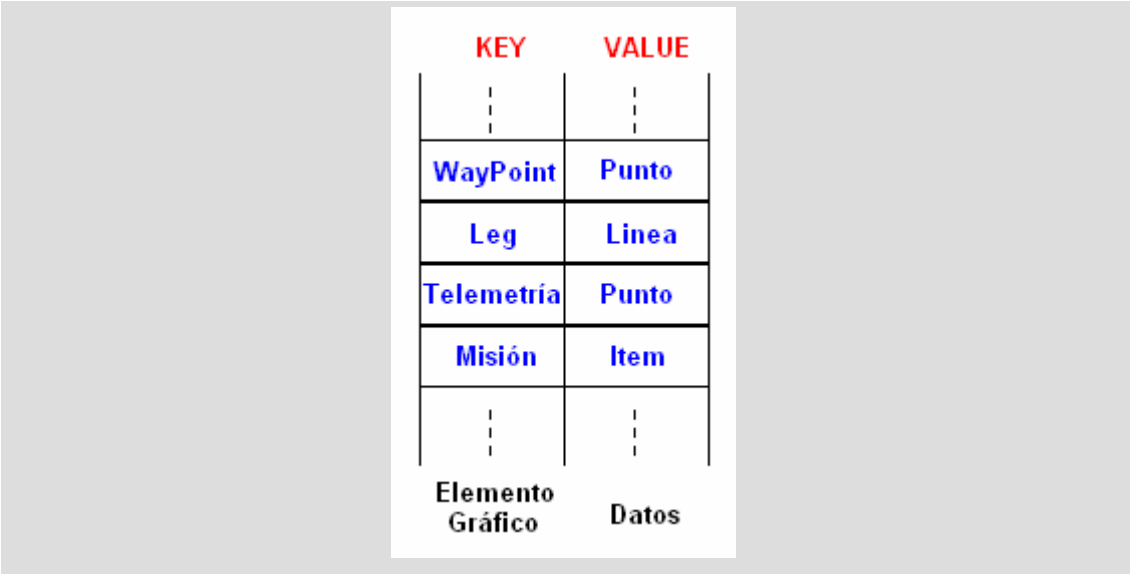
El principal problema es que una vez cargados los elementos gráficos al container, si se desea extraerlos y analizarlos, estos ahora no son ni waypoints ni legs, sino puntos y líneas, tal y cómo habían sido cargados.

Una posible solución es buscar la coordenada en la que se encuentra dicho elemento e ir a buscarla al plan de vuelo. Es un método muy ineficiente ya que se debe recorrer constantemente el plan de vuelo. Otra es utilizar la tabla de hash.

5.1.6. Container y Hash

La tabla de hash permite guardar una relación de elementos. Se van introduciendo elementos de dos en dos a la tabla de hash. La tabla los va guardando, un elemento en una lista (key), y el otro en otra (value), pero guarda la relación entre ellos. A la hora de quitarlos, se podrá buscar un elemento tanto en una lista como en otra. De esta manera, al encontrar dicho elemento, se podrá preguntar por el que está relacionado con este.

Figura 23. Diagrama Explicativo de Tabla Hash



Se ahorra mucho tiempo en búsquedas. La mecánica será la siguiente:

- Antes de introducir un elemento gráfico al container, se introducirá al hash este mismo elemento y su correspondiente (al que representa)
- Al quitar un elemento del container, también se quitará del hash.

De esta manera todos los elementos que estén en el container, también estarán en el hash, y se podrá buscar su pareja.
Por lo tanto los pasos intermedios implementados desde pasar el ratón por un elemento ubicado en el mapa, a enseñar su información serán los siguientes:

- Identificar el elemento seleccionado en el container.
- Buscar este elemento gráfico en la tabla de hash.
- Una vez encontrado, preguntar por su pareja.
- Ya se tiene los datos del elemento. Ahora se enseñan.

5.1.7. Pintar el seguimiento de la misión

Es más sencillo que el plan de vuelo ya que sólo se pintan los paquetes de telemetría recibidos. Los “legs” de telemetría no son más que una línea pintadas entre dos paquetes de telemetría consecutivos.

Se debe utilizar el hash, ya que como se ha dicho antes, todos los elementos del container deben estar en la tabla de hash. Así se podrá recuperar la información de telemetría.

El último paquete de telemetría recibido será pintado de forma distinta a los anteriores. Este paquete se enviará al container como un ítem con forma de avión, ya que representa la última información enviado por él. Además se pintará el vector velocidad normalizado. Este vector indica dónde estará el avión en un segundo después.

Una vez se reciba otro paquete, el anterior debe ser pintado como un paquete de telemetría viejo, y la línea que representa el modulo de la velocidad borrada, para así pintar la nueva infamación.

5.1.8. Paneles de información

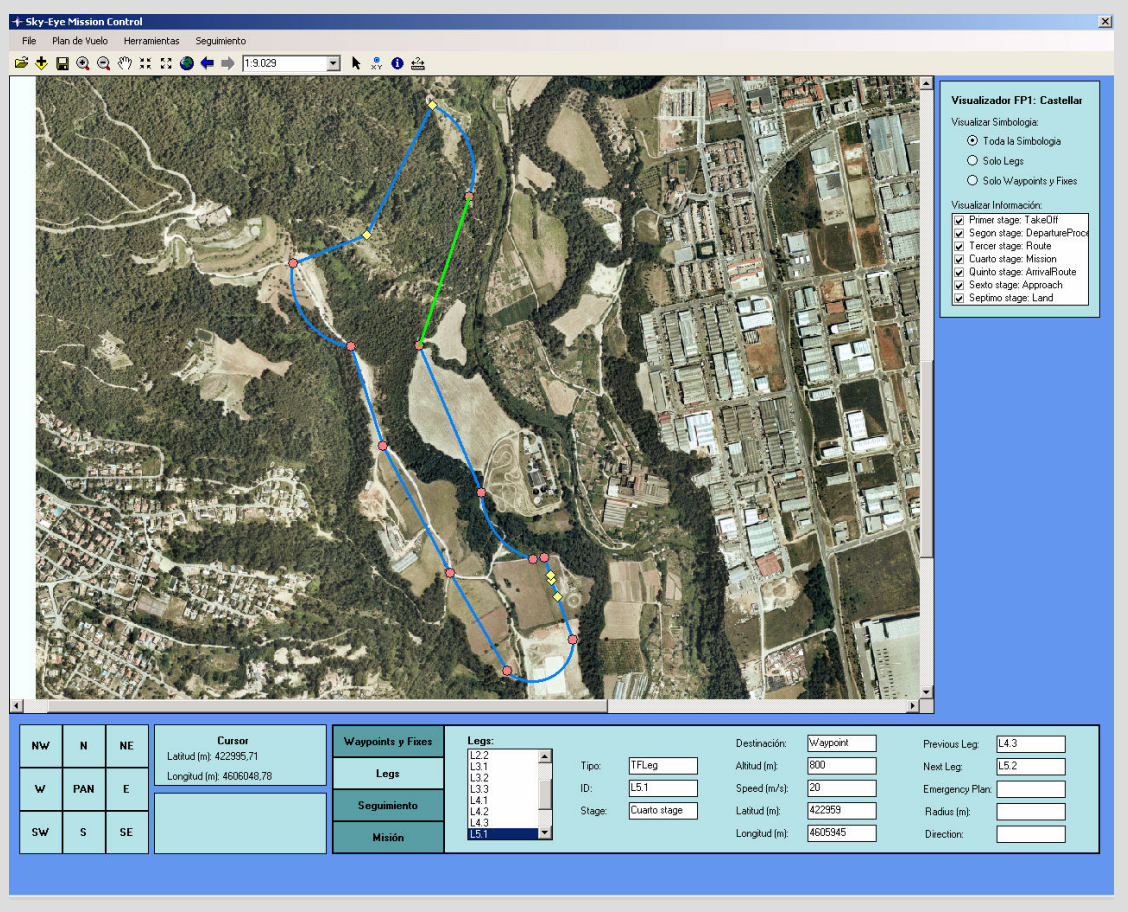
Además de visualizar la información de forma visual en el mapa, se cargarán unos paneles que muestren de forma enumerativa lo que se está viendo en el mapa.

Hay paneles con información de Waypoints y Fixes, Legs, Telemetría o seguimiento.

De esta manera, a demás de poder consultar la información pasando el ratón sobre el elemento, también se podrá hacer clic sobre la lista el elemento que queremos ver. Por lo tanto existirá un lazo entre la información que está en el mapa y la que hay en las listas. Cuando un elemento se selecciona, tanto de las listas como en el mapa, el elemento en las listas (su nombre) quedará “selected”, y en el mapa (su representación gráfica) se pintará de otro color.

En la Figura 24 se puede observar el resultado de las funciones descritas anteriormente.

Figura 24. Visualización de un Plan de Vuelo Pintado

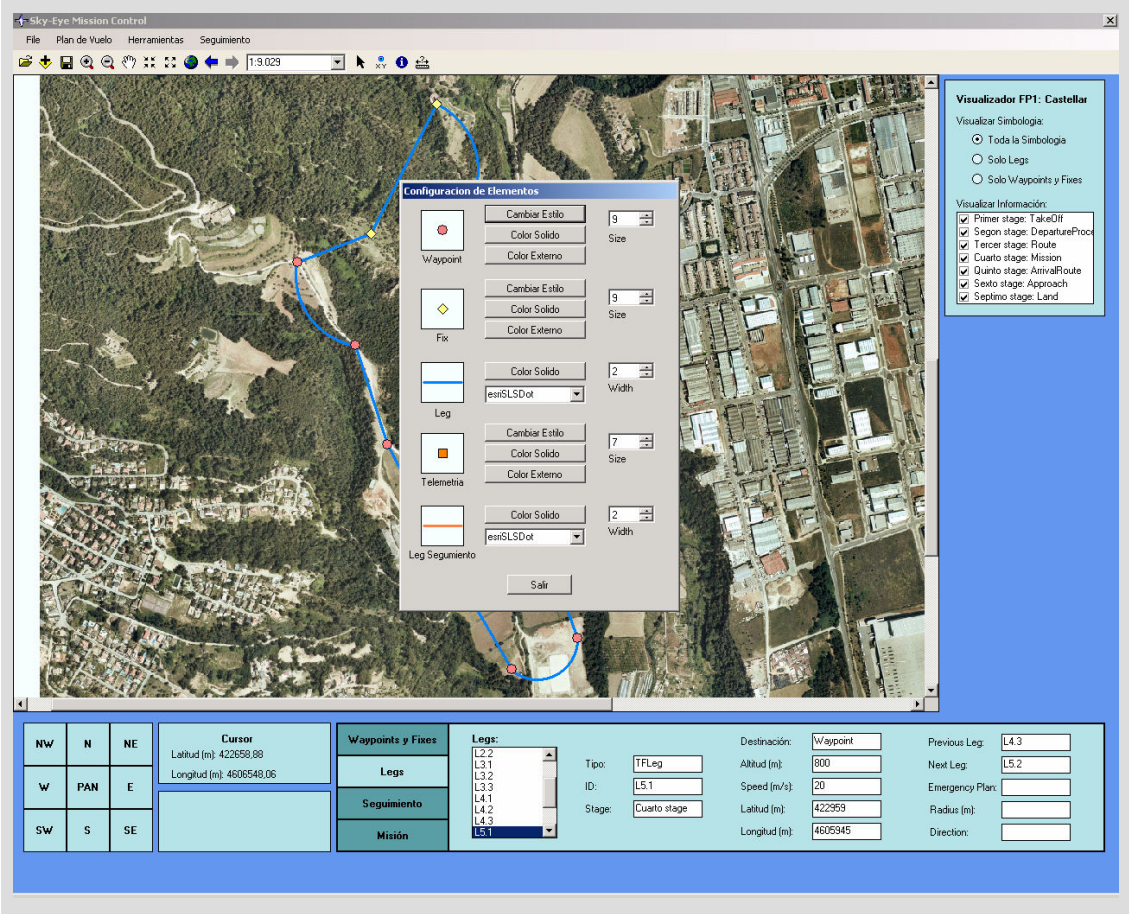


5.2. Panel de configuración de elementos

En la Figura 25 se puede ver el diseño final del gestor para configurar elementos. Como se puede apreciar en la figura aparece una vista previa y al lado unos botones para cambiar la configuración.

Cabe destacar que los datos son actualizados en la estructura de clases. Para pintar las vistas previas se accede a estas clases, a pesar de que se pueda acceder con variables locales, y así pintar los elementos tal y como se hace en el programa principal.

Figura 25. Centro de la imagen: Gestor para configuración de Elementos



5.3. Pruebas individuales

Se han realizado pruebas individuales de cada módulo. Es decir, antes de integrar dos funciones, cada una de ellas ha debido pasar una prueba de correcto funcionamiento. De esta manera se consigue asegurar que los dos bloques funcionan correctamente, y en caso de haber un error general, este mismo error ha debido de ser causado por un conector entre ambos bloques.

Un claro ejemplo es el pintado de un plan de vuelo: primero se verifica la función pintar punto y pintar línea. Una vez funcione se pintará un stage, para luego pasar a pintar todos los stages, es decir, el plan de vuelo.

5.4. Pruebas de integración

Tal y como se ha comentado al comienzo de este TFC, la aplicación está compuesta por el trabajo de tres estudiantes. Una vez acabadas todas las etapas se debe realizar la integración de los sistemas y las pruebas.

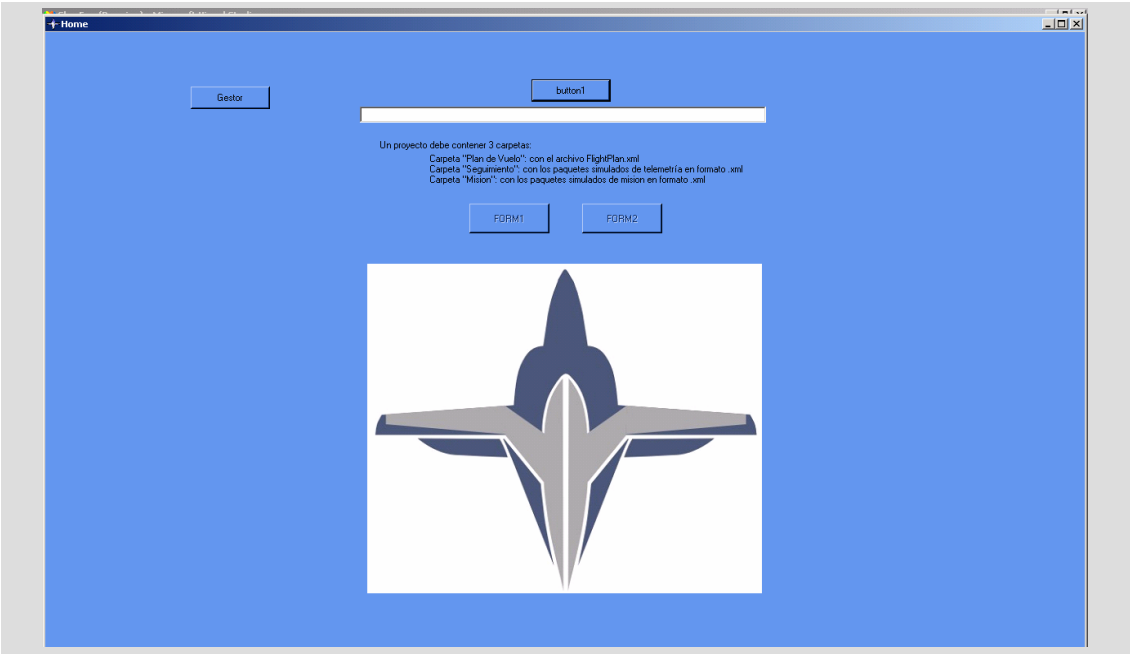
Al integrar la parte de lectura de plan es de vuelo y estructura de datos, con la de interfaz de usuario, se debe comprobar que todos los campos se lean correctamente. Es decir, que el camino intermedio entre pintar la información y llegar hasta ella, funcione.

Los principales problemas se encuentran al agregar las librerías y hacer las referencias. Todas las variables que se creaban desde el formulario encargado de la visualización, ahora deben ser buscadas en las clases.

Existe más de un formulario. Uno de los problemas es compartir información entre ellos. Para conseguirlo se crea una clase nueva llamada Exchange Area con información común a todos. Todos los formularios que necesiten información externa deberán acceder a ella.

Una vez integrados estos dos sistemas, se debe integrar el de georeferenciación. Se crea una pantalla de presentación con botones para acceder tanto a un formulario como al otro. En esta pantalla se pedirá el path de la carpeta de trabajo, para así no tener que pedirla en ambos formularios.

Figura 26. Vista de la Presentación de la Aplicación



Tanto un formulario como el otro pueden trabajar independientes. Por lo tanto pueden estar los dos abiertos o no.

CAPÍTULO 6. BALANCE

En los inicios del proyecto se estimaron una serie de costos, tanto económicos como de tiempo. Este capítulo intentará contrastar dichas previsiones con los valores reales, y justificar las diferencias.

6.1. Contraste de Planificación

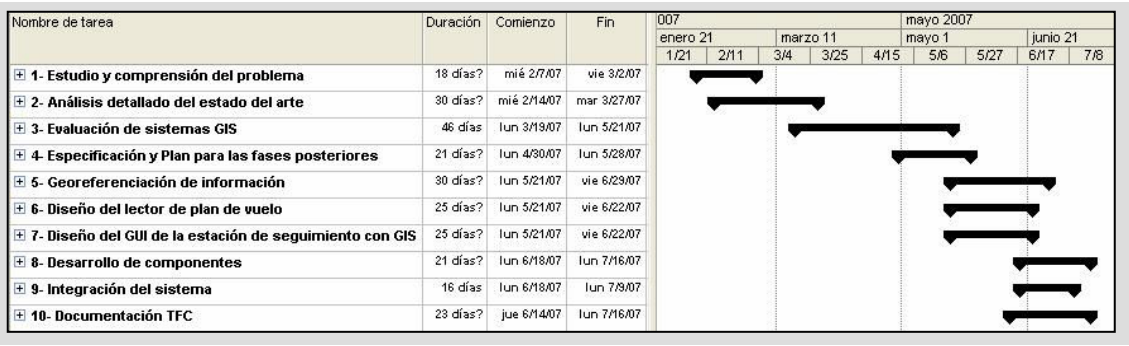
Las tareas de cada parte del proyecto han variado en duración. El trabajo de análisis ha sido mucho más largo del esperado, reduciendo horas de programación. Se ha intentado dar un poco más de horas a la parte de documentación con el fin de poder explicar con más detalle los problemas sufridos.

La principal causa de esta variación ha sido la adaptación a los sistemas GIS. Se ha invertido mucho tiempo en conocer los software que habían en el mercado, y evaluar sus diferencias.

Una vez acabada esta evaluación y seleccionado el software de ESRI, hubo que dedicar mucho tiempo a conocerlo un poco más a fondo. La curva de aprendizaje no resultó ser tan corta, y el principal motivo es la complejidad del entorno GIS.

En el diagrama que se enseña en la Figura 27 se puede ver la duración final de las tareas:

Figura 27. Diagrama de tareas Final



El diagrama refleja claramente la larga duración del análisis, y cómo se ha acortado tiempo en el desarrollo e integración.

Se pudo ahorrar tiempo de programación gracias a la buena elección hecha tras el análisis. Aunque las herramientas (SDK de ESRI) no son baratas, sí son muy eficientes. Estas herramientas fueron bien utilizadas y de esta manera se consiguieron prácticamente los mismos fines con menos tiempo.

Uno de los principales motivos es que una vez se tiene el SDK, también se tiene la plataforma base hecha, y programar nuevas funciones no cuesta tanto.

6.2. Presupuesto Final y Justificaciones

A pesar que se hayan cumplido gran parte de los objetivos propuestos a tiempo (ver capítulo 7), cabe destacar que la descompensación de horas entre tareas de analistas y programadores ha generado un incremento en sueldos. Esto se debe a que los analistas cobran siete euros más la hora que los programadores.

En el presupuesto inicial no se contaba con el software de desarrollo. Es un gasto que como se explicó en su momento no estaba previsto, y al final ha resultado significativo a la hora de calcular los costes finales.

RECURSOS HUMANOS INDIVIDUAL	€/hora	Horas		CosteTotal
		inicial	final	
Analista Informático	32	130	190	6.080 €
Programador	25	235	300	7.500 €
Documentador	20	80	120	2.400 €
TOTAL		445	610	15.980 €

RECURSOS HUMANOS GRUPO TRABAJO	€/hora	Horas		CosteTotal
		inicial	final	
Analista Informático	32	390	636	20.352 €
Programador	25	705	822	20.550 €
Documentador	20	240	382	7.640 €
TOTAL		1335	1840	48.542 €

INFRAESTRUCTURA	Cantidad	coste unidad	coste total
Estación de Trabajo	3	1.000 €	3.000 €
Licencia Windows XP	3	159 €	477 €
Licencia Sharepoint	1	2.000 €	2.000 €
Licencia Microsoft Office	3	450 €	1.350 €
Licencia Microsoft Visual Studio 2005	3	679 €	2.037 €
Licencia Development Kit ESRI	3	1.750 €	1.750 €
Materiales de oficina varios			500 €
		TOTAL	11.114 €
		COSTE TOTAL	59.656 €

El coste total ha aumentado en comparación con la previsión hecha al comienzo del proyecto.

Esto es debido a que se ha aumentado en un mes la duración del proyecto, por tanto se han generado más gastos de personal.

A este factor se le suma el gasto en software de desarrollo: Microsoft Visual Studio 2005 y el Development Kit de ESRI.

Cabe destacar que estos precios son una estimación de lo que costará este proyecto en un ámbito profesional. El proyecto SKYEYE se está desarrollando con muchos estudiantes y profesores.

Con respecto a la infraestructura no se ha adquirido toda. De hecho se han utilizado muchos ordenadores personales. Las licencias de software han sido facilitadas por la universidad. En caso de haber sido compradas, probablemente se hubiese comprado una licencia para estudiantes con el fin de reducir gastos.

Se puede encontrar más información sobre el tiempo y esfuerzos empleados en el proyecto en el ANEXO B.

CAPÍTULO 7. CONCLUSIONES

7.1. Revisión de objetivos

"Un objetivo es el conjunto de resultados cualitativos que se propone alcanzar a través de determinadas acciones". Con esta definición, que se puede encontrar en algunos diccionarios, entendemos que un objetivo significa marcarse una meta e intentar conseguirla. Puede que no la consigamos. Podemos aprovechar esta situación y analizar porqué no se han conseguido. De esta manera probablemente sepamos los impedimentos que han causado este retraso, y así tenerlo en cuenta para un próximo intento.

Podemos estar satisfechos con la aplicación. Prácticamente hemos conseguido todos los objetivos propuestos, ya que tenemos un visualizador de planes de vuelo y seguimiento eficientes.

7.1.1. Objetivos Individuales

La aplicación consigue cargar varios tipos de cartografías y ortofotos y visualizarlas correctamente. Una vez cargado el plan de vuelo, se puede visualizar los elementos que éste contiene con independencia y comodidad. De la misma manera, se ha solucionado de forma adecuada la visualización de los paquetes de telemetría.

Además de enseñar esta información se ha conseguido mostrar de forma clara la información que contienen los elementos en la pantalla. El usuario de esta aplicación puede representar gráficamente una gran cantidad de información, y al mismo tiempo puede distinguirla y clasificarla.

Se ha tenido en mente constantemente el usuario de esta aplicación. Por ello se han creado herramientas, que si bien no son de gran importancia para el cumplimiento de los objetivos puramente técnicos, sí lo son para que el usuario tenga más comodidad. Un claro ejemplo es la selección de color y estilo de todos los componentes del mapa.

Como se ha comentado en otros puntos, el gran problema técnico que ha alargado el proyecto ha sido el mostrar toda esta información en capas. Esto hubiese ayudado a estructurar mejor la información sin la necesidad de muchas líneas de código. Sin embargo se ha encontrado una muy buena alternativa que resuelve de forma eficiente este problema. No se debe interpretar como un factor negativo, sino al contrario. Es de interés saber cuál es la mejor manera de hacer un trabajo, y en caso de ser posible realizarlo. Pero hay que tener en cuenta el gran valor que supone haber tenido una alternativa en un campo que es apenas conocido, y con un plazo de tiempo ciertamente ajustado.

Por consiguiente se puede concluir que se han cumplido gran parte de los objetivos. Si bien no en todos los casos se han realizado de la manera más efectiva, se han buscado alternativas igual de eficientes de cara al usuario.

7.1.2. Objetivos grupo INFINIFISH

Han sido muchos los motivos por los que no se han cumplido algunos objetivos. Esta aplicación se ha comenzado a programar cuando aún no estaba acabado el trabajo del grupo ICARUS requerido por nuestro proyecto. Esto ha provocado un gran vacío a la hora de diseñar ciertas partes de la aplicación. También ha creado retrasos. Un claro ejemplo han sido los cambios en el modelo conceptual del plan de vuelo en plena época de desarrollo, provocando un trabajo adicional de adaptación.

Las partes más perjudicadas han sido la visualización y almacenaje de información en capas, y la ortonormalización. Las dos tienen en común su complejidad a la hora de programar. Si bien las librerías que ofrece ESRI son muy efectivas, si no se tiene una buena base de GIS, se puede desperdiciar esfuerzos, y como es el caso, no conseguir dichos objetivos a tiempo.

No obstante, de los objetivos propuestos el más notorio es la ortonormalización de fotografías. Se han conseguido buenas aproximaciones que si bien no nos aportan toda la información necesaria, si ofrecen gran parte de ella. Para más información sobre este tema puede verse el TFC de Miriam González.

Todos los objetivos conseguidos hacen de esta aplicación un buen prototipo de inspiración y futuro desarrollo para el proyecto SKY EYE.

7.2. Conclusiones

Al ser el primer gran proyecto que realizo, teniendo en cuenta su importancia, y el tiempo que se le ha dedicado, se pueden sacar muchas conclusiones al respecto. Están las que describen el proyecto en sí, las que concluyen una etapa de trabajo en grupo, y las personales. En este apartado se intentarán plasmar todas ellas.

7.2.1. Conclusiones Técnicas

El proyecto consta de varias partes. Entre ellas está la investigación y documentación, el diseño y desarrollo, y por ultimo la implementación y pruebas.

Investigar y documentar supone tiempo. Realizar bien esta tarea nos ha facilitado mucho trabajo en las posteriores etapas del proyecto. Era muy necesario invertir algo más de un mes en buscar información sobre GIS ya que debíamos conocer un poco este sector a de por sí complejo. Probablemente si

pudiésemos haberle dedicado más tiempo, podríamos haber solucionado algunos problemas que han aparecido posteriormente con más facilidad.

Para comenzar a desarrollar debemos conocer bien nuestros objetivos y requerimientos (marcados por un cliente) y las herramientas que tenemos para conseguirlos. Se ha tenido presente al usuario en todo momento. Los Casos de uso han sido de gran ayuda para hacer esta aplicación fácil de usar.

Podemos concluir que ha sido complicado desarrollar al principio. Programar con un SDK conlleva mucho estudio, ya que no empezamos de cero, sino que aprovechamos una serie de herramientas que ya están creadas. Estas herramientas no se pueden utilizar de cualquier manera. En el caso del development kit de EDN, cada vez que buscábamos crear algo nuevo, antes debíamos documentarnos. Esto supone un tiempo que se debió tener en cuenta al principio.
Por otro lado hay que ser concientes que estas mismas herramientas pueden resolvernos muchos problemas y ahorrarnos mucho tiempo en futuros desarrollos.

7.2.2. Conclusiones de grupo

Al ser un proyecto en grupo, tanto a nivel de INFINIFISH como de SKY EYE, debemos aceptar la idea que compartimos tiempos, ideas y esfuerzos. Todo nuestro trabajo será utilizado por otros, y viceversa. Por ello tenemos que intentar que este trabajo sea lo más claro posible. Un claro ejemplo es el código tecleado: éste debe estar muy bien estructurado y documentado.

Si bien nos marcamos objetivos, y los vamos cumpliendo, no siempre se consiguen los resultados que buscábamos, ya que dependemos del trabajo de más gente. Estos objetivos se cambian, y por tanto se cambia la forma en la que trabajábamos. Por ello tenemos que desarrollar una gran capacidad de adaptación a las nuevas circunstancias. Y cuanto más rápida sea esta adaptación, mejor.

Con respecto a la planificación y el cumplimiento de ella, cabe mencionar que hay que ser muy estrictos con lo que se propone y lo que se acaba haciendo. Una serie de retrasos acumulados pueden suponer un gran problema al final, sobre todo si se tiene una fecha límite.

7.2.3. Conclusiones personales

Además de haber hecho un trabajo final de carrera, he aprovechado el tiempo empleado para colaborar en un proyecto de cierta envergadura, formarme como ingeniero, y aprender de lecciones que no se dan en las clases.

El TFC es la última tarea académica que desarrolla el ingeniero. Es la conexión entre la carrera y el mundo laboral, y también se puede aprender cosas con ello. Como por ejemplo estructurar y planificar un proyecto de mediana

duración, plantearse objetivos y buscar la manera de cumplirlos, redactar una memoria, etc.

Considero fundamentales estas lecciones ya que pueden ser habituales en el mundo laboral.

El trabajo en equipo ha supuesto un gran adelanto en el proyecto ya que ha resuelto dudas y problemas de forma más rápida.

Y por último, hemos conseguido desarrollar una aplicación que cumple los objetivos propuestos inicialmente. Es una aplicación que posiblemente sirva de esqueleto para crear otras nuevas con nuevas y mejores funciones.

7.3. Líneas de Trabajo futuro

El proyecto se ha marcado objetivos para diseñar un prototipo de aplicación. Para conseguir un buen funcionamiento y mayor contribución al proyecto SKY-EYE se deben desarrollar las técnicas aplicadas en este prototipo para conseguir una aplicación madura en contenidos.

Siguiendo esta idea, existen varias líneas de trabajo que se podría seguir.

- Mejorar el aspecto visual utilizando capas temáticas en lugar de contenedores de elementos.
- Ortonormalizar las fotos recibidas y poder crear un mosaico de ortofotos con dicha información.
- Desarrollar la técnica de consulta de paquetes de telemetría (hasta ahora lo que hay hecho es un emulador).
- Modificar el modelo conceptual de los paquetes recibidos (telemetría o misión) en función de los que marquen las otras partes de SKY EYE.
- Conectar la aplicación a una red, por ejemplo ETHERNET, con el fin de extender las capacidades del proyecto. De esta manera se podrá controlar varios UAV a la vez si es necesario, que realicen la misma o distinta misión.

Es de esperar que con el transcurso del tiempo se vayan generando nuevas necesidades. Por ello es interesante tener en cuenta esta lista, con el fin de recordar los objetivos a corto plazo (los que se proponen al acabar el prototipo), y poder ir añadiendo los nuevos.

Es de mi interés seguir involucrado en el desarrollo de esta aplicación. No descarto esta opción ya que mi línea de trabajo futuro será continuar con la Ingeniería Superior en Telecomunicación.

7.4. Estudio de la ambientalización

En este proyecto no hace falta indagar mucho para encontrar aspectos que ayuden a preservar o contribuir de alguna forma al medio ambiente.

El proyecto está enfocado en este momento a la prevención de incendios forestales, y a una buena y efectiva contribución con su extinción. Al estar Cataluña en una zona de pocas lluvias, mucha vegetación, y altas temperaturas, cualquier ayuda es buena para poder disminuir o erradicar este problema que padece la sociedad.

Si desarrollamos un poco más esta idea, todas las funciones que ahora mismo hacen los helicópteros de vigilancia y soporte de estación las harán los UAV. De esta manera el gasto de gasolina, la contaminación acústica, y contaminación que los vehículos con hélices produzcan quedarán reducidas.

Este es sólo un aspecto. Probablemente una vez conseguido dicho objetivo se busquen otros motivos con el fin de contribuir con las reservas de nuestro mundo.

BIBLIOGRAFÍA

[1] GIS – SIG Sistemas de información geográfica
< http://www.gabrielortiz.com/ >

[2] NASA Unmanned Aerial Vehicles
< http://uav.wff.nasa.gov/ >

[3] Colección de software GIS gratuito
< http://www.gabrielortiz.com/index.asp?info=063 >

[4] Recopilación de teoría de GIS: Core Currículum in GIS de la N.C.GIA
< http://www.gabrielortiz.com/index.asp?info=071 >

[5] Geographic Information System (GIS): libro digital
< http://www.gabrielortiz.com/index.asp?info=078 >

[6] The source for GIS and mapping software
< http://software.geocomm.com/raster >

[7] Espinosa Oscanoa, Domingo “Procesamiento Digital de Ortofotos”.

[8] García León, Josefa y Cuarteto Sáez, Aurora “Compraración de procesos de rectificación y ortoproyección mediante fotogrametría terrestre digital”.

[9] E. Freeville “INTEGRA Metrics & Methodologies Efficiency Algorithm Assessment” 2002

[10] UAV Ground Control Stations
< http://www.uavm.com/uavsubsystems/groundcontrolstations.html >

[11] Institut Cartográfic de Catalunya
< http://www.icc.es/portal/ >

[12] Fire Paradox
< http://www.fireparadox.org/ >

[13] Comets Project “Fire Monitoring”
< http://www.comets-uavs.org/applications/fire.shtml >

[14] Fira GlobalGeo
< http://www.globalgeobcn.com/ >

[15] ESRI – The GIS Software Leader
< http://www.esri.com/ >

[16] ESRI Developer Network
< http://edn.esri.com/ >

[17] gvSIG
< http://www.gvsig.gva.es/ >

[18] GRASS GIS
< http://grass.itc.it/ >

[19] Miramon: Geographic Information System and Remote Sensing Software
< http://www.creaf.uab.es/MiraMon/ >

[20] QUANTUM GIS
< http://community.qgis.org/ >

[21] .Net Compact Framework, 2005
< http://msdn2.microsoft.com/en-us/library/f44bbwa1.aspx >

[22] MSDN Home Page España
< http://msdn2.microsoft.com/es-es/default.aspx >