Proyecto fin de carrera

Software para la gestión de un proyecto de datos LiDAR en un sistema de información geográfica.

Daniel López Sánchez
Tutor: Miguel Sánchez López

Valencia septiembre 2010

Índice general

1. Introducción **1**

1.1. Propósito 1
1.2. La Empresa DIELMO 3D S.L. 2
1.3. Metas y Objetivos 2
1.4. Contexto del Proyecto 5
1.5. Dependencias del proyecto 7

2. La Tecnología LiDAR **9**

2.1. Introducción a la Tecnología LiDAR 9
2.2. Generalidades 10
2.3. Ventajas e Inconvenientes 13
2.4. Aplicaciones 15

3. Antecedentes **25**

3.1. Introducción 25
3.2. Sistema de información geográfica 26
3.2.1. Aplicaciones de los SIG. 31
3.2.2. Modelos de representación. 43
3.2.3. Flujo de trabajo en un sistema de información geográfica. 49
3.2.4. Infraestructuras de Datos Espaciales 51
3.2.4.1. WFS (Web Feature Service): 51
3.2.4.2. WMS (Web Feature Service): 52
3.2.4.3. WCS (Web Coverage Service): 53
3.2.5. Historia de los sistemas de información geográfica. 53
3.2.6. Sistemas de información geográfica en el mercado 55
3.3. El sistema de información geográfica gvSIG 55
3.3.1. Formatos soportados 59
3.3.2. Herramientas disponibles 60
3.3.3. La interfaz de gvSIG 63
3.3.4. Proyectos y Documentos propios de gvSIG . . . 64

3.3.4.1. Vista 65
3.3.4.2. Tabla 69
3.3.4.3. Mapa 71
3.3.5. Extensión geoBD (gestor de bases de datos) . . 75
3.3.5.1. La extensión espacial de PostgreSQL. PostGIS 77
3.3.6. Extensión de Geoprocesamiento 80
3.3.7. Libreria Java Topology Suite 88
3.3.8. Libreria JFreeChart 90
3.4. Software LiDAR 91
3.4.1. Productos de Terrasolid 91
3.4.1.1. TerraScan 92
3.4.1.2. TerraSlave 94
3.4.1.3. TerraMach 95
3.4.1.4. TerraModeler 96
3.4.1.5. TerraPhoto 98
3.4.1.6. TerraSurvey 100
3.4.1.7. TerraPipeNet 101
3.4.1.8. TerraPipe 101
3.5. Otras aplicaciones relacionadas 101
3.5.1. SEXTANTE 101

4. Planificación y evaluación de costes 111
4.1. Consideraciones previas 111
4.2. Planificación inical 112
4.3. Seguimiento 114

5. Análisis 117
5.1. Introducción 117
5.2. Arquitectura de gvSIG 120
5.2.1. Introducción 120
5.2.2. Andami. 122
5.2.2.1. Funcionalidad. 124
5.2.2.2. Plugins y extensiones. 128
5.2.2.3. El fichero config.xml 130
5.2.2.4. Extensiones 138
5.2.2.5. Gestión de ventanas de Andami 142
5.2.2.6. Servicios a los plugins 152

5.2.3. La librería libFMap 160
5.2.3.1. MapControl 162
5.2.3.2. MapContext 164
5.2.3.3. Layers 173
5.2.3.4. DataSources y Drivers 179
5.3. Análisis funcional 180
5.3.1. Introducción 180
5.3.2. Casos de uso 180

6. Diseño **215**
6.1. Introducción 215
6.2. Contexto 216
6.3. Diseño del ducumento LiDAR 217
6.4. Diseño del plugin 218
6.4.1. Diagramas de secuencia 222
6.5. Diseño interfaces 222

7. Implementación **231**
7.1. Introducción 231
7.2. Eclipse 231
7.3. Implementación del documento LiDAR 232
7.3.1. Persistencia en gvSIG. 234
7.4. Implementación del plugin 237
7.4.1. Dependencias del proyecto 237
7.4.2. El fichero config.xml 238
7.4.3. La extensión de carga de capas 240
7.4.4. La extensión de selección del proyecto activo . . 245
7.4.5. La extensión de visualización de gráficas longitudinales 247
7.4.6. La extensión de conversión de .LAS a .XYZ . . 252
7.4.7. La extensión de conversión de .XYZ a .LAS . . 256
7.5. Implementación de algoritmos 258

8. Pruebas, resultados y rendimiento **263**
8.1. Introducción 263
8.2. Pruebas de rendimiento 265

9. Conclusiones **269**

9.1. Síntesis del trabajo 269
9.2. Colaboraciones 271
9.3. Perspectivas de futuro y mejoras. 273

10. Anexos **277**
10.1. Manual de usuario DielmoOpenLidar 277

A. Acrónimos **313**

B. Licencia **315**

Bibliografía (Libros y artículos) **325**

C. Agradecimientos **327**

Índice de figuras

1.1. Logo de DIELMO 3D S.L. 2
1.2. Logo de gvSIG. 6
1.3. Logo de SEXTANTE. 7

2.1. Esquema de funcionamiento de la tecnología LiDAR. . 11
2.2. Penetración del láser en la vegetación. 12
2.3. Comportamiento del rayo láser en diferentes superficies. 13
2.4. Esquema de la adquisición de datos mediante el láser. . 15
2.5. Esquema del comportamiento del rayo láser en la vegetación. 18
2.6. Ejemplo de la penetración del láser en la vegetación. . 19
2.7. MDT y MDS de un bosque con vegetación intensa de Galicia. 20
2.8. Imágenes del proyecto de Segovia realizado por DIELMO 3D S.L. 21
2.9. Envolvente de calados máximos por inundación costera en el Término municipal de Oliva 23

3.1. Componentes de un sistema de información geográfica. 26
3.2. Sistema de información geográfica. 27
3.3. Esquema de representación multicapa de un SIG. . . . 28
3.4. Esquema de las funciones GIS. 31
3.5. Interfaz de un sistema de gestión de flotas. 32
3.6. SIG con una capa vectorial visible del catastro. 33
3.7. Sistema basado en gvSIG gisEIEL. 34
3.8. Aplicación GIS para la dministración local. Diputación de Jaén. 34
3.9. Aplicación GIS para la explotación de racursos naturales. 35
3.10. Aplicación GIS mostrando cálculos de áreas de influencia. 35
3.11. Servicio WMS del Camino de Santiago. 36
3.12. Aplicación GIS para el estudio de la salud pública. . . 37
3.13. Aplicación GIS para situaciones de catástrofe. 38

3.14. Aplicación GIS para el estudio de la criminalidad. . . . 39
3.15. Aplicación de gvSIG en un estudio relativo al monitoreo de incendios. (Quintas jornadas gvSIG) 40
3.16. Salida gráfica de la aplicación de estudio relativo al monitoreo de incendios. 41
3.17. Aplicación GIS parala enseñanza de geografía. 41
3.18. Nokia sport tracker. GPS tracker aplicado a prácticas deportivas. 42
3.19. Aplicación Asignet para la gestión de redes de fibra óptica. 42
3.20. Esquema de los modelos ráster y vectorial. 43
3.21. Sistema de coordenadas geográficas que utiliza las dos coordenadas angulares latitud (norte o sur) y longitud (este u oeste) para determinar las posiciones de la superficie terrestre. 45
3.22. Sistema de coordenadas UTM (Universal Transverse Mercator). 47
3.23. Dispositivos para la adquisición de datos espaciales. . . 50
3.24. Logo del OGC. 51
3.25. Infraestructuras de datos espaciales. 52
3.26. Tabla comparativa de diferentes GIS (fuente wikipedia). 56
3.27. Logo GNU GPL. 58
3.28. Menú añadir capa de gvSIG. 59
3.29. Menú añadir capa de gvSIG. 60
3.30. Herramientas de navegación. 61
3.31. Interfaz principal de gvSIG. 64
3.32. Tipos de documentos en gvSIG. 65
3.33. Documento vista en gvSIG. 65
3.34. Tabla de contenidos de una vista. 66
3.35. Ventana de visualización de una vista. 66
3.36. Localizador de una vista. 67
3.37. Documento tipo tabla en el gestor de proyectos. 69
3.38. Tabla en gvSIG. 70
3.39. Herramientas del menu Tabla. 71
3.40. Documento tipo mapa en el gestor de proyectos. 72
3.41. Documento tipo mapa. 73
3.42. Símbolos de norte. 74
3.43. Opciones asociadas a un mapa. 74

3.44. Plantilla de un mapa. 75
3.45. Interfaces de la extensión geoBD. 76
3.46. Certificación OGC. 78
3.47. Área de influencia de 4 puntos. 81
3.48. Enlace espacial. 82
3.49. Enlace espacial. 83
3.50. Diferencia. 84
3.51. Intersección. 84
3.52. Unión. 85
3.53. Convex Hull. 85
3.54. Dissolve. 86
3.55. Juntar. 87
3.56. Juntar. 87
3.57. Reproyeccion. 88
3.58. Métodos de análisis espacial de JTS. 89
3.59. Distintos tipos de gráficas de JFreeChart. 90
3.60. Visualización de un corte longitudinal de datos LiDAR 93
3.61. Esquema de TerraSlave 95
3.62. La línea de vuelo y diferentes pasadas del láser útiles para calibrar los datos. 96
3.63. Datos antes y después de ser ajustados. 97
3.64. Datos LiDAR antes y después de eliminar el solape entre distintas pasadas del avión. 98
3.65. Modelos digitales de TerraModeler. 98
3.66. Corrección de color de ortofotos. 99
3.67. Diseño de la red de tuberías. 102
3.68. Visualización de la red de tuberías en 3D. 103
3.69. Gestor de extensiones de SEXTANTE. 103
3.70. Extensión de SEXTANTE. 106
3.71. Modelador gráfico de SEXTANTE. 107
3.72. Ejecución por lotes de SEXTANTE. 107
3.73. Línea de comandos de SEXTANTE. 108

4.1. Diagrama de Gantt de la planificación del proyecto. . . 115

5.1. Aparición de errores en las distintas fases de desarrollo de software. 119

5.2. Coste de corrección de errores producidos en las distintas fases de desarrollo de software. 119
5.3. Esquema de la arquitectura de gvSIG. 120
5.4. Diagrama de subsistemas de gvSIG. 121
5.5. Esquema de la arquitectura de gvSIG. 123
5.6. Descripción de los bloques funcionales de Andami. . . . 125
5.7. Jerarquía de etiquetas válidas de plugin-config.xml. . . 131
5.8. Diagrama de las clases implicadas en la creación de extensiones. 139
5.9. Diagrama de las principales clases implicadas en la gestión de ventanas. 143
5.10. Clases principales de CorePlugin y clases relacionadas de Andami. 145
5.11. Diagrama que muestra las principales clases relacionadas con el gestor de ventanas. 149
5.12. Diagrama de bloques de FMap. 161
5.13. Visión estructural de la librería libFMap. 162
5.14. diagrama de clases simplificado de MapControl. 164
5.15. diagrama de clases de MapContex. 166
5.16. diagrama de clases simplificado de ViewPort en MapContext. 170
5.17. Una vista de gvSIG, donde se indica la dimensión del área disponible para visualizar las capas. 173
5.18. La vista de la Imagen 1, sobre la que se ha cargado dos capas. El usuario selecciona un área de interés, esta será el extent. Se recalculará el nuevo adjustedExtent. . 173
5.19. Área de interés seleccionada, adaptada a las dimensiones disponibles, ampliado el ancho seleccionado, por mantener una proporción al aspecto del área disponible de visualización. 174
5.20. Diagrama de clases que muestra la relación de FLayerStatus con las capas de libFMap. 174
5.21. Diferentes estados de las capas. 175
5.22. relación entre las clases e interfaces del grupo Layers. . 177
5.23. Diagrama simplificado que muestra los principales tipos de capas. 178
5.24. Diagrama de bloques de Subdriver. 179

5.25. Diagrama inicial de subsistemas. 181
5.26. Diagrama de contexto del subsistema gestión. 205
5.27. Diagrama de contexto del caso de uso definición de un proyecto LiDAR. 206
5.28. Diagrama de contexto del caso de uso definir datos LiDAR. 207
5.29. Diagrama de contexto del caso de uso definir zonas de control. 207
5.30. Diagrama de contexto del caso de uso definir cartografía ráster. 208
5.31. Diagrama de contexto del caso de uso definir cartografía vectorial. 208
5.32. Diagrama de contexto del caso de uso definir area de interés. 209
5.33. Diagrama de contexto del caso de uso definir shape de bloques. 209
5.34. Diagrama de contexto del caso de uso definir ruta. . . . 210
5.35. Diagrama de contexto del caso de uso definir resolución. 210
5.36. Diagrama de contexto del caso de uso definir solape. . . 211
5.37. Diagrama de contexto del caso de uso definir tamaño máximo de edificio. 211
5.38. Diagrama de contexto del caso de uso definir productos. 212
5.39. Diagrama de contexto del caso de uso definir leyenda de clasificación. 212
5.40. Diagrama de contexto del caso de uso definir operador. 213
5.41. Diagrama de contexto del caso de uso definir fichero log. 213

6.1. Diagrama de contexto del proyecto. 216
6.2. LidarDocument dentro de gvSIG. 217
6.3. LidarDocument dentro de gvSIG. 219
6.4. Diagrama de secuencia creación de un documento LiDAR. 220
6.5. Diagrama de clases de una nueva extensión en gvSIG. . 221
6.6. Diagrama de extensiones del plugin. 222
6.7. Diagrama de secuencia cargador de capas avanzado. . . 223
6.8. Diagrama de secuencia selección del documento LiDAR activo. 224
6.9. Diagrama de secuencia visualización de gráficas longitudinales. 224

6.10. Diagrama de secuencia conversor avanzado de .LAS a .XYZ. 225
6.11. Diagrama de secuencia conversor de .XYZ a .LAS. . . . 226
6.12. Interfaz herramienta de conversión de .xyz a .las 227
6.13. Interfaz herramienta de conversión de .las a .xyz 227
6.14. Interfaz herramienta de carga avanzada de capas. . . . 228
6.15. Interfaz recogida de datos del documento LiDAR. . . . 228
6.16. Interfaz para definir leyenda en el documento LiDAR. . 229
6.17. Interfaz panel recogida de parámetros del documento LiDAR. 229

7.1. Entorno Eclipse. 232
7.2. Logo de Eclipse. 233
7.3. Documentos en gvSIG junto al documento LiDAR desarrollado. 233
7.4. Diagrama de clases simplificado de LidarDocument. . . 234
7.5. Vista del plugin en el Package Explorer de Eclipse. . . 237
7.6. Dependencias del plugin con otros proyectos. 238
7.7. Descripción de la extensión en el menú preferencias. . . 242
7.8. Descripción de la extensión en el menú preferencias. . . 242
7.9. Herramienta de cargado automático de capas. 243
7.10. Botón de la extensión de selección del documento LiDAR activo. 246
7.11. Herramienta de selección de franjas. 249
7.12. Perfil de un valle. 253
7.13. Iconos .LAS a .XYZ. y .XYZ a .LAS 255
7.14. Herramienta conversión de .LAS a .XYZ. 255
7.15. Herramienta conversión de .XYZ a .LAS 256

9.1. Plan de vuelo PNOA 2009. 274

CAPÍTULO 1

Introducción

1.1 Propósito

Se propone el diseño e implementación de una aplicación para la gestión de un proyecto de datos LiDAR en un sistema de información geográfica. LiDAR o Light Detection and Rangin consiste en la medición de distancias con láser, en nuestro caso con aplicaciones cartográficas. El objetivo de esta aplicación es dotar al usuario final de un conjunto de herramientas que le permitan la gestión estructurada de un proyecto de datos LiDAR dentro de un sistema de información geográfica como gvSIG. La aplicación dispondrá de elementos que permitan al usuario la visualización, análisis, clasificación y conversión entre formatos de los datos y proporcionará una serie de herramientas de cálculo para dar soporte a la elaboración de productos finales como por ejemplo MDT[1]o MDS[2] a partir de datos LiDAR.

[1]MDT: Modelo Digital del Terreno: estructura numérica de datos que representa la distribución espacial de una variable cuantitativa y continua, como puede ser la temperatura, la cota o la presión atmosférica. En particular, cuando la variable a representar es la cota o altura del terreno se denomina Modelo Digital de Elevaciones o MDE.

[2]MDS: Modelo Digital de Superficie: modelo que describe la superficie del terreno incluyendo las estructuras (como por ejemplo edificios). Para ayudar a entenderlo sería como echar una sábana por encima del terreno.

1.2 La Empresa DIELMO 3D S.L.

Figura 1.1: Logo de DIELMO 3D S.L.

DIELMO 3D S.L. (Digital Elevation Models) es una empresa de servicios cuyas oficinas se encuentran Xirivella (Valencia). Especializada en la generación de Modelos Digitales del Terreno con diferentes resoluciones adaptándose a todo tipo de necesidades y aplicaciones. Desde el año 2003 realiza trabajos de adquisición y procesado de datos LiDAR en España. El procesado de dichos datos lo realiza con software propio que viene desarrollando desde hace 6 años para utilizarlo internamente.

Esta empresa es representante en España de Infoterra Ltd. Infoterra es uno de los principales proveedores de datos LiDAR del mundo, con una experiencia en cuatro continentes y es líder europeo en la distribución de imágenes de satélite. De aquí que DIELMO 3D también distribuya datos procedentes de sensores térmicos aerotransportados, imágenes de satélite y datos hiperespectrales.

Mediante DIELMO 3D, S.L. se me ofreció la oportunidad de realizar prácticas en empresa durante 6 meses y el Proyecto Final de Carrera a la vez que entraba en contacto con la tecnología LiDAR. El periodo de prácticas comenzó el 6 de octubre de 2008 y finalizó el 6 de abril de 2009.

1.3 Metas y Objetivos

Satisfacer al conjunto de usuarios finales de datos LiDAR es nuestro principal objetivo.

"El problema que hay actualmente es que no hay prácticamente nada que permita tratar datos LiDAR al usuario estándar. Hay 4 o 5 personas o empresas en España que tengan software comprado que permita hacerlo. Pero fuera de ahí, ni siquiera los usuarios que pagan para que les hagan un vuelo LiDAR pueden abrir y visualizar los datos

para hacer un control de calidad. Entonces ahí hay un mercado bastante grande que es el usuario final, el usuario estándar de SIG". Jose Carlos García – Director gerente de DIELMO (Jornadas gvSIG).

Se debe distinguir entre los distintos tipos de usuarios de dichos datos. Por un lado tras la adquisición de datos, la empresa encargada de tal fin necesita realizar, como se detalla más adelante, una serie de ajustes de dichos datos adquiridos antes de su entrega a la entidad contratadora. Por otro lado esta entidad contratadora de la adquisición de datos puede tener una serie de necesidades distintas a las del grupo anterior como puede ser la elaboración de Modelos Digitales del Terreno (MDT) o Modelos digitales de Superfice (MDS) para realizar estudios de inundabilidad, planificación urbanística etc. entre muchas otras. En capítulos sucesivos se dan a conocer más necesidades que pueden tener estos usuarios de datos LiDAR. Así dentro de los usuarios finales de nuestra aplicación software distinguimos dos grupos:

-Proveedores de datos LiDAR: Empresas encargadas de la toma de datos LiDAR. Tras la toma de datos se requiere una serie de herramientas que permitan el ajuste de los datos que el láser ha registrado. Es decir, antes de su entrega o su utilización final tienen que ser preprocesados y ajustados. Más adelante se detalla en qué consiste este ajuste.

-Destinatarios finales de dichos datos: En este grupo se encuentran empresas o entidades que contratan una toma de datos LiDAR al grupo anterior para la realización de estudios de naturaleza variada como por ejemplo estudios hidráulicos, inventarios forestales, estimación de daños causados por catástrofes, planificación urbanística, mantenimiento de líneas eléctricas etc. En posteriores capítulos se especifican muchas de las aplicaciones más usuales.

El objetivo principal de nuestra aplicación será el de proveer a ambos grupos de usuarios de las herramientas necesarias para la gestión de un proyecto LiDAR. Entre estas distinguiremos varios grupos:

-Herramientas de gestión: Permitirán la definición de los datos de entrada del proyecto. Entre estos datos distinguimos: ficheros LiDAR en

formatos LAS o BIN, ortofotos, puntos de control, resolución, área de interés, shape de bloques(ESRI Shapefile .SHP es un formato vectorial de almacenamiento digital donde se guarda la localización de los elementos geográficos y los atributos asociados a ellos), capas vectoriales (por ejemplo de edificios vectorizados, puentes, límites de ríos, etc), definición de leyendas de clasificación, etc. Por lo tanto un proyecto LiDAR puede manejar una gran volumen de datos de entrada de distintos tipos. Es de gran utilidad, por consiguiente, proporcionar al usuario final un conjunto de herramientas de gestión que le permitan tener bien definidos y organizados los datos de entrada, para su posterior utilización. Más adelante se detallan todas las herramientas que conforman este grupo que permiten definir los elementos de un proyecto LiDAR.

-Herramientas de visualización: Permitirán hacer una exploración visual de los datos. La pretensión en este proyecto es poder dotar al sistema de información geográfica de la capacidad de visualización de una gran cantidad de datos, del orden de cientos de GigaBytes de datos LiDAR brutos (nube de puntos irregulares en formato LAS o BIN) . Para el desarrollo de este tipo de herramientas ha sido muy importante la eficiencia de los algoritmos y la estrategia aplicada para su implementación precisamente por el gran volumen de datos con los que se pretende trabajar. Como se explica posteriormente se han implementado varios tipos de visualizaciones en función de la altura, clasificación e intensidad. Por otro lado se ha dotado a la aplicación de una herramienta de visualización avanzada para la representación de perfiles longitudinales de los datos LiDAR haciendo uso de la información de la altura que éstos datos nos proporcionan.

-Herramientas de clasificación: Permitirán al usuario hacer una clasificación manual de los datos LiDAR definidos en el proyecto. La clasificación permite diferenciar la naturaleza de los datos. Podemos clasificar edificios, vegetación alta, media o baja, agua, suelo, y aplicar a esta clasificación diferentes leyendas que el usuario tendrá la posibilidad de definir y ampliar.

-Herramientas de cálculo: Permitirán al usuario un procesado de los

datos para distintas finalidades como la obtención de productos finales como MDT, MDS, etc. Consistirán en una colección de algoritmos de procesado, análisis, conversión de formatos, etc. La intención es dotar a la comunidad de usuarios de la capacidad de extender las funcionalidades del proyecto mediante la facilidad de implementar nuevos algoritmos rápidamente. De esta forma implementar nuevas funcionalidades para cálculos con datos LiDAR se reduce únicamente a la implementación de un nuevo algoritmo y su incorporación a este proyecto como se explica más adelante.

1.4 Contexto del Proyecto

La idea de DIELMO 3D S.L. inicialmente era la de comercializar el software que estaba desarrollando y vender licencias para su utilización, viendo en su software una oportunidad de negocio ya que éste en muchos aspectos era más eficiente que mucho software comercial existente. Sin embargo, DIELMO 3D S.L. entró en contacto con la Conselleria de Infraestructuras y Transporte de la Comunidad Valenciana conociendo el proyecto gvSIG. GvSIG es un sitstema de información geográfica de la Conselleria de infraestructuras y Transporte de la comunidad Valenciana. Se trata de un proyecto open source. En este momento DIELMO se decide por todo lo contrario. Se propone implementar dentro de gvSIG el software que estaba utilizando internamente. Así se pretende hacer un software libre para trabajar con datos LiDAR dentro de un sistema de información geográfica libre como es gvSIG.

Con el desarrollo de este software libre para el manejo de datos LiDAR, DIELMO 3D pretende acercar el uso de la tecnología LiDAR a los usuarios SIG estándar y a la comunidad científica, con el objetivo de extender su uso. Además, actualmente se tiende a que cada vez haya más datos LiDAR disponibles que cubran grandes extensiones del territorio y de aquí a unos años toda esta información será libre para que cualquiera pueda acceder a ella. Por ejemplo, hay disponibles datos LiDAR de todo el País Vasco y se pueden conseguir a través de la Diputación Foral de Guipúzcoa y del Servicio de Cartografía del Departamento de Medio Ambiente y Ordenación del Territorio de Go-

bierno Vasco. Una pequeña muestra de estos datos va a ser utilizada durante este proyecto para ir mostrando las herramientas que se han implementado.

Figura 1.2: Logo de gvSIG.

En 2003 la Conselleria de Infraestructuras y transportes sacó a concurso el desarrollo e implantación de un nuevo programa para el manejo de información geográfica (SIG). IVER Tecnologías de la Información es la empresa ganadora del concurso que lleva el peso del desarrollo. Así nace gvSIG. Algunas de las características a destacar de este proyecto son: portable, modular, de código abierto, sin licencias, sujeto a estándares. El proyecto gvSIG cuenta con una infraestructura de colaboración sólida. Un proyecto que colabora con gvSIG es el proyecto SEXTANTE o Sistema EXTremeño de ANálisis TErritorial. SEXTANTE es una extensión de gvSIG. Consiste en una librería de algoritmos de análisis geoespacial. El objetivo principal de SEXTANTE es crear una plataforma que facilite tanto el uso como la implementación de estos algoritmos. SEXTANTE facilita la tarea del programador del algoritmo quien no tiene que implementar la interfaz gráfica, pues ésta es generada por el propio SEXTANTE. Por esta razón se entiende que SEXTANTE puede ser un complemento ideal para este proyecto permitiéndonos implementar a nosotros y a la comunidad de colaboradores nuevas funcionalidades y algoritmos al proyecto LiDAR de manera sencilla. En posteriores capítulos se detallan aspectos técnicos de SEXTANTE y gvSIG y se aclara en qué consiste la implementación de nuevos algoritmos de SEXTANTE.

Nos encontramos trabajando con una tecnología en expansión con un amplio abanico de aplicaciones como es la tecnología LiDAR. Dentro de un sistema de información geográfica libre, con una comunidad de usuarios y desarrolladores en crecimiento y con un alto componente en I+D+I como es gvSIG. Con otros proyectos y extensiones a nuestro alcance que pueden ser de gran utilidad para propósitos variados que facilitan la extensión de funcionalidades como es SEXTANTE. Y con

el impulso que supone colaborar en un proyecto como gvSIG que avanza rápidamente y cuenta cada vez con más aceptación a nivel global.

1.5 Dependencias del proyecto

Trabajar dentro de un marco de colaboración implica una serie de ventajas y posibilidades de progreso evidentes. Del mismo modo aparecen una serie de limitaciones y dependencias con las tecnologías con las que se está trabajando. El sistema de información geográfica gvSIG cuenta con un grupo de supervisión y normalización, a modo de organismo vertebrador que permite dar una cierta coherencia a los nuevos desarrollos.

Como se menciona en el punto anterior en el proyecto está totalmente ligado a gvSIG y hace uso de SEXTANTE. Por otra parte hay distintos formatos LiDAR. Estos factores son muy importantes a tener en cuenta. La evolución del proyecto va a depender muy estrechamente de los cambios que en ellos se produzcan. Tanto futuras versiones de gvSIG y de SEXTANTE como la aparición de nuevos formatos LiDAR implicarán la necesidad de adaptación del proyecto.

Figura 1.3: Logo de SEXTANTE.

CAPÍTULO 2

La Tecnología LiDAR

2.1 Introducción a la Tecnología LiDAR

La medición de distancia con láser (Light Detection and Ranging o LiDAR) resulta de la combinación de diversos elementos. La finalidad del sistema es la detección y medición de distancias desde el espacio aéreo a través de la luz láser con fines cartográficos. Este capítulo describe las características fundamentales de la tecnología LiDAR, su funcionamiento,aplicaciones y ventajas e inconvenientes de su uso frente a otras tecnologías.

2.2 Generalidades

El sistema LiDAR aerotransportado está revolucionando la adquisición de datos digitales de elevación para diferentes aplicaciones del ámbito de la cartografía como por ejemplo estudios de inundabilidad, elaboración de mapas etc. LiDAR es una alternativa de garantía para la generación de Modelos Digitales del Terreno (MDT) de calidad en cuanto a precio y tiempo de elaboración.

La medición se realiza por medio de un telémetro [1] que va montado en una plataforma aerotransportada como por ejemplo aviones de ala fija o helicópteros para realizar un barrido sobre una zona del terreno mediante la emisión del láser. El rebote en el terreno del láser es captado por el sensor lo que permite determinar la distancia entre los dos puntos. Los pulsos láser que son reflejados desde tierra son recibidos por la óptica del sensor y se transforma en información electrónica. Por medio de unos medidores de intervalos de tiempo (TIMs), se mide con alta precisión el tiempo que cada pulso tarda en viajar hasta el suelo y regresar al dispositivo emisor. Conociendo la velocidad de propagación del láser en la atmósfera es posible determinar la distancia desde el emisor a la superficie. Esto junto con sistemas de posicionamiento GPS [2] permite tener la información de la altura y la posición de los puntos que se han registrado.

El comportamiento del pulso láser sobre los distintos tipos de superficie es diferente:

- En una **superficie sólida** (suelo, edificios, etc.) el rayo láser refleja sin problemas y vuelve al avión. En este caso la diferencia entre el primero y el último pulso pequeña (del orden del error del instrumento).

- En el **agua** el rayo láser es absorbido y no retorna al sensor. Por lo tanto no se obtiene ninguna información.

[1] dispositivo capaz de medir distancias de forma remota.
[2] GPS: Global Positioning System o Sistema de Posicionamiento Global que permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con una precisión hasta de centímetros, usando GPS diferencial, aunque lo habitual son unos pocos metros

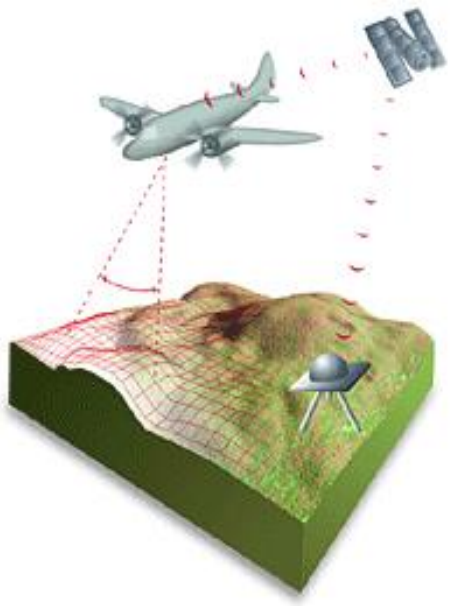


Figura 2.1: Esquema de funcionamiento de la tecnología LiDAR.

- En la **vegetación** parte del pulso es reflejado al chocar contra ella en su parte superior pero al tratarse de una superficie no sólida parte de él penetra hasta el suelo que también lo refleja. De esta manera se puede determinar la distancia entre ambos rebotes o lo que es lo mismo la frondosidad o altura de la vegetación. Mediante LiDAR por lo tanto se puede determinar la altura del suelo en zonas incluso con vegetación densa.

Para el correcto posicionamiento de los puntos medidos en el terreno se utilizan dos técnicas combinadas:

- INS (Sistema de Navegación Inercial): mide la orientación del sensor. Mide los ángulos con una precisión de 0.001 grados lo que permite compensar cualquier cambio brusco de dirección al que pueda verse sometido el sensor en el avión a causa de turbulencias y conocer las coordenadas exactas del punto que se ha medido.

- GPS diferencial: permite establecer la posición exacta del sensor.

Un operador es el encargado de controlar durante el vuelo las funciones del sistema LiDAR a través de un interfaz de operador (OI).

Figura 2.2: Penetración del láser en la vegetación.

Durante la adquisición de datos el instrumento genera y emite estrechos pulsos de luz infrarroja y un espejo de barrido dirige los pulsos en dirección perpendicular a la línea de vuelo. El movimiento del espejo cubre a ambos lados de la dirección de vuelo, en función del ángulo especificado por el operador. El avance del avión cubre en la dirección del vuelo produciéndose un barrido del terreno.

Durante la misión de adquisición de datos GPS en el avión se graba la información del sistema de orientación en una memoria de alta velocidad, mientas que el ángulo y la posición del espejo de barrido se guardan en la unidad de colección de datos principal.

Tras la fase de adquisición de datos, en el post-procesado se combina toda esta información con el GPS diferencial para la obtención de precisa de las coordenadas x, y, z (datos de latitud, longitud y altura del terreno) para cada uno de los pulsos. La adquisición de los datos posicionales se realiza a intervalos definidos. El equipo con el que trabaja la empresa utiliza una frecuencia de adquisición de datos de 33KHz.

Los datos resultantes LiDAR son una red de puntos muy densa con

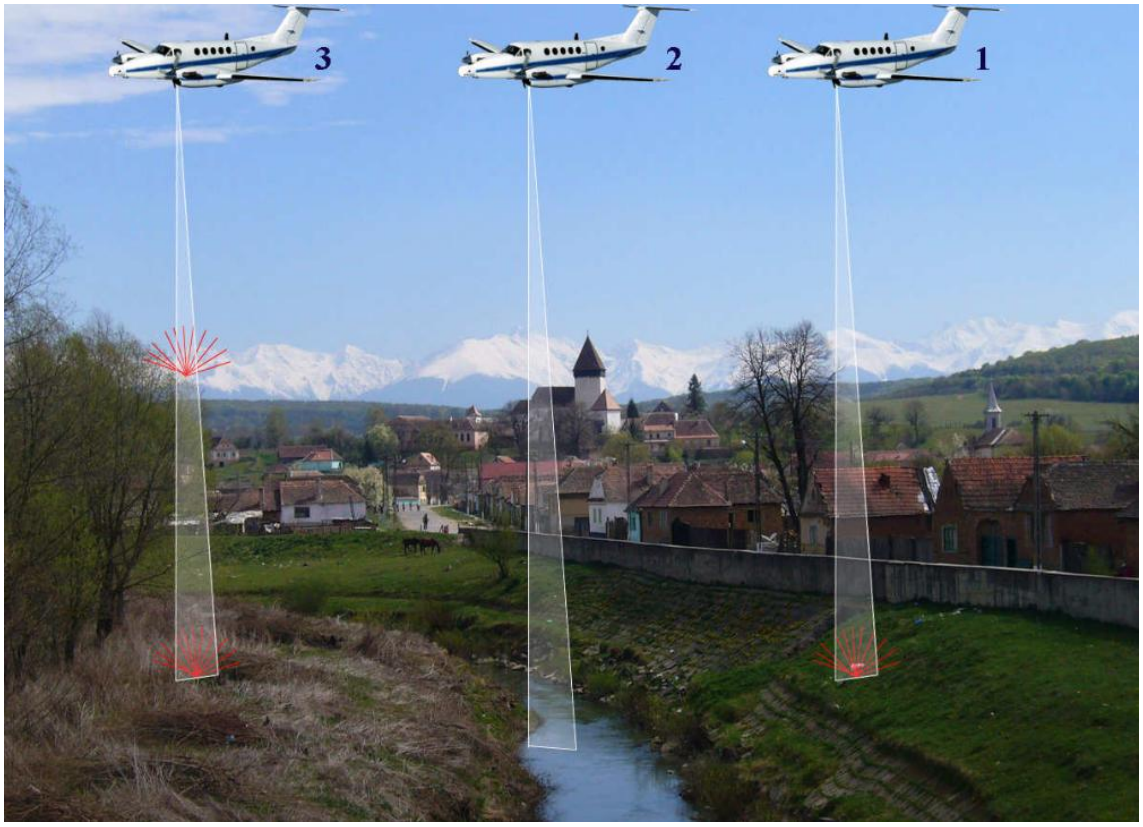


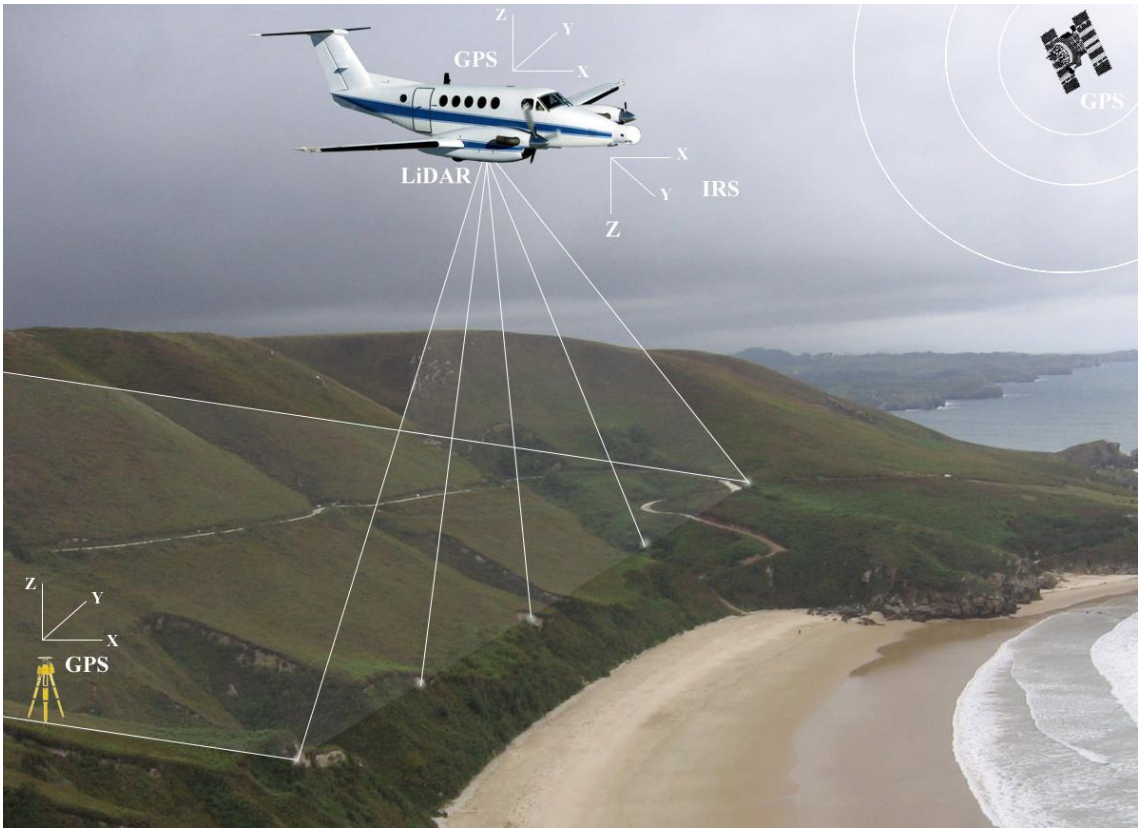
Figura 2.3: Comportamiento del rayo láser en diferentes superficies.

una densidad variable de puntos por metro cuadrado en función de la altura del vuelo y del ángulo de barrido del láser. En general la precisión de los datos LiDAR esta determinada por la altura del vuelo, el diámetro del rayo láser, la calidad de los datos GPS / INS y los procedimientos de postprocesamiento. En un vuelo estándar a 850 m. de altura se obtiene una precisión horizontal de 0.424 m. (altura de vuelo/2000). La precisión horizontal es mejor de 15 cm. dentro de una desviación estándar.

El post-procesado de los datos del avión y las medidas de GPS en tierra se completan en tierra. Es este procesado de datos uno de los objetos de estudio de este proyecto.

2.3 Ventajas e Inconvenientes

Algunas de las ventajas e inconvenientes de esta tecnología se citan a continuación:



Ventajas:

- Los vuelos de las misiones LiDAR pueden realizarse durante el día o la noche y con condiciones meteorológicas adversas como con bruma, al contrario que con la fotografía aérea.
- El procesado de los datos se acelera ya que la disposición de los datos es de forma digital e inmediata.
- En zonas costeras y forestales de encuentran más fácilmente los puntos de suelo.
- El coste del procesado de datos LiDAR es inferior al de la fotogrametría [3] convencional para alcanzar el mismo nivel de precisión.
- Posibilidad de obtener gran densidad de puntos por metro cuadrado.

[3] La fotogrametría es la ciencia o técnica cuyo objetivo es el conocimiento de las dimensiones y posición de objetos en el espacio, a través de la medida o medidas realizadas a partir de la intersección de dos o más fotografías, o de una fotografía y el modelo digital del terreno correspondiente al lugar representado, el cual ha de ser realizado anteriormente por intersección de dos o más fotografías.

(a) Primer pulso de ida desde el avión hacia el suelo

(b) Primer y último pulso de regreso al avión tras rebotar en el terreno.

Figura 2.4: Esquema de la adquisición de datos mediante el láser.

Inconvenientes:

- El coste del equipo de adquisición es alto, superando el millón de euros por lo que la amortización del mismo es larga.

- No se obtiene una imagen georeferenciada de calidad. En ocasiones se requiere la realización de otro vuelo para la obtención de ortofotos de la zona.

- La precisión del espejo disminuye con la altura del vuelo. Al tener que realizar vuelos más bajos es necesaria la realización de más líneas de vuelo con lo que aumenta el coste económico de la adquisición de datos.

- Con la aparición de viento o turbulencias es posible la aparición de errores en el sistema inercial.

2.4 Aplicaciones

La tecnología LiDAR nos permite la obtención de un MDT de alta precisión permitiendo el uso de éste en multitud de aplicaciones de

distinta naturaleza. Las aplicaciones más utilizadas por DIELMO 3D S.L. son las del ámbito de la cartografía, como son la clasificación (edificios, vegetación, suelo, etc) , creación de escenarios virtuales en 3 dimensiones, cartografía para estudios de inundabilidad, etc...

Son muchas las aplicaciones que se le pueden dar a los productos obtenidos por la tecnología LiDAR, destacando entre otras:

- Aplicaciones forestales.
- Cartografía.
- Cartografía para estudios de inundabilidad.
- Modelos urbanos en 3D.
- Generación de mapas de ruido.
- Escenarios virtuales en 3D.
- Actualización muy precisa de la línea costera.
- Modelado de líneas de alta tensión.

A continuación se detallan todas estas aplicaciones:

- **Aplicaciones forestales:**

  Mediante el uso combinado de cámaras digitales, LiDAR aerotransportado y láser terrestre es posible modelizar el medio forestal parametrizando los principales indicadores del inventario forestal. La industria de la madera y la de la selvicultura han realizado ahorros de tiempo y de coste en el empleo de los servicios de

MDT mediante la tecnología LiDAR. Determinar la elevación del terreno debajo de las ramas de los árboles siempre ha resultado una dificultad muy importante para la fotogrametría clásica. Esto se produce debido a que por debajo de la vegetación densa es difícil apreciar el terreno con claridad. Por el contrario, con el uso de la tecnología LiDAR, se consigue que alguno de los pulsos del láser penetre hasta el suelo entre la vegetación (ver figura 2.5). Dependiendo de la altura a la que se realice el vuelo , el ángulo de penetración del láser y el número de pasadas obtendremos más información por metro cuadrado. En proyectos realizados por DIELMO 3D S.L. Se ha llegado a obtener un MDT de 4 puntos por metro cuadrado. La exactitud de los inventarios forestales LiDAR es muy superior al método tradicional, los resultados se obtienen en menos tiempo y la información capturada es utilizable para múltiples propósitos de gestión, conservación y defensa del bosque. Mediante la tecnología LiDAR y muchos años de investigación se tiene la capacidad en la actualidad incluso de contar y cubicar uno por uno todos los árboles de un bosque. La capacidad del LIDAR de penetrar en la vegetación y obtener el MDT y el MDS, permite supervisar y determinar los volúmenes de madera sobre bases regulares. Combinado con las imágenes digitales infrarrojas, se crea una solución valiosa que hace posible determinar las condiciones y crecimiento que apoye el desarrollo y recolección de madera.

Un proyecto realizado por DIELMO en Lugo consistió en la realización de un inventario forestal a partir de los datos LiDAR procesados por DIELMO, generando un fichero en formato TXT con el número de árboles, la posición y la cota de cada uno de ellos. El índice de penetración del ALTM (equipo de Infoterra en los vuelos realizados para DIELMO) es a menudo mejor del 30 % en bosques coníferos, de hojas caducas y mixtos.

Podemos obtener a partir del MDT del LiDAR y de la clasificación de vegetación los siguientes productos que juegan un papel fundamental en el ámbito forestal:

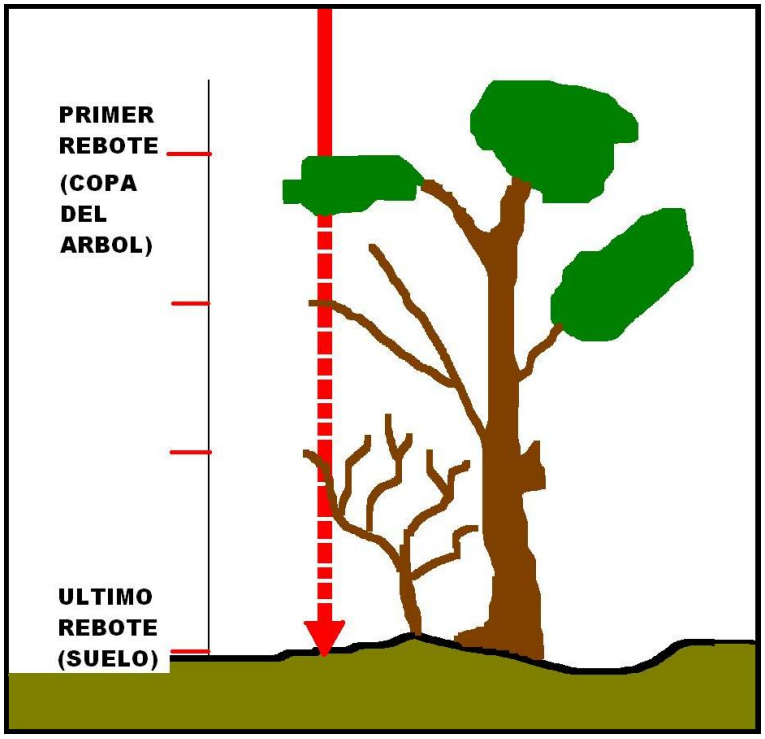


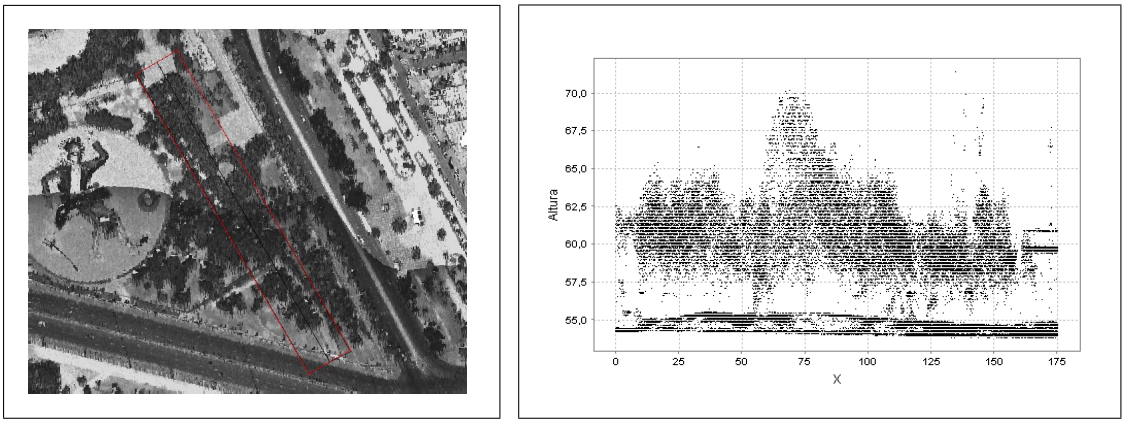
Figura 2.5: Esquema del comportamiento del rayo láser en la vegetación.

- Generación de un MDT preciso por debajo de la vegetación.
- Cálculo preciso de la altura de cada árbol.
- Cálculo preciso de la posición de cada uno de los árboles.
- Cálculo de los volúmenes, LAI (Índice de Área Foliar), etc.

- **Cartografía:**

  Partiendo del MDT obtenido con el LiDAR y de la clasificación de edificios y vegetación se podría realizar cartografía digital de gran precisión, disminuyendo considerablemente tanto los tiempos de realización como los precios. Esta información junto con datos de una ortofoto (la cual se puede corregir a partir del MDT obtenido) o imagen actualizada de la zona permiten generar mapas temáticos. En noviembre de 2004 se realizó la adquisición y el procesado de datos LiDAR en una hoja entera del MTN.25 de la zona de Segovia para el IGN dando como resultado un MDT con 1 metro de resolución y mejor de 15 cm. de precisión en altura.

  En la figura 2.8 se muestran imágenes de la zona de Segovia con los resultados ofrecidos por DIELMO al IGN (Instituto Geográfico



(a) Imagen LiDAR de intensidades del Gulliver (Río Turia, Valencia)

(b) Perfil longitudinal de la arboleda de la figura a

Figura 2.6: Ejemplo de la penetración del láser en la vegetación.

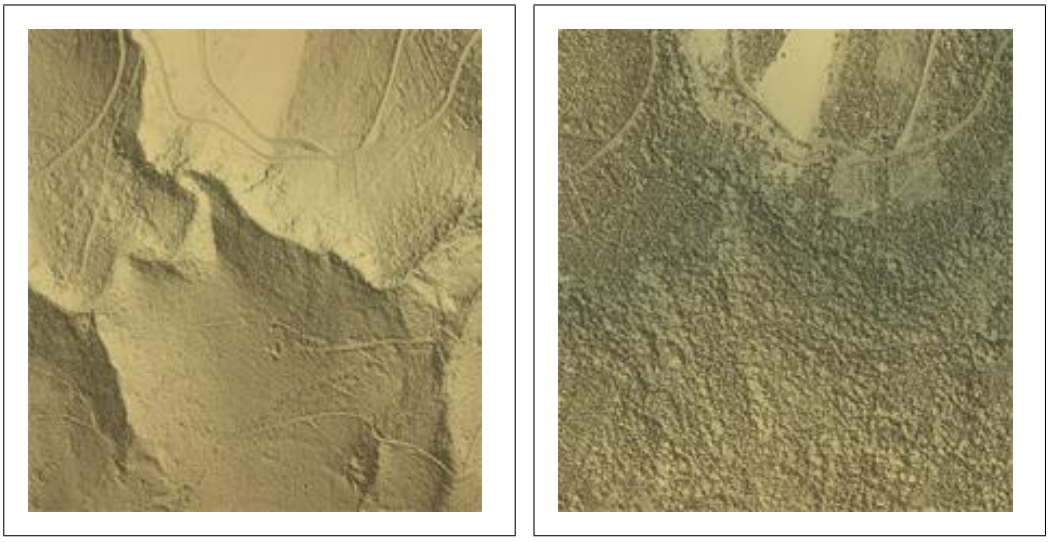
Nacional).

- **Cartografía para estudios de inundabilidad:**

  A partir de un MDT de alta resolución se puede realizar un estudio de inundabilidad. Se deben eliminar los puentes presentes puesto que el agua realmente está pasando por debajo de ellos. Esto evita que actúen como una presa. Para la realización de un estudio de este tipo hay que llevar a cabo otros previamente como:

  - Estudio geomorfológico: aporta información sobre la génesis y desarrollo de las inundaciones que pueden afectar a la zona de estudio. Así sus resultados sirven para entender el funcionamiento hidráulico de la zona inundable. En base al MDT se elabora un mapa geomorfológico que indica la evolución de los caminos de los flujos desbordados. Para ello se obtiene a partir del LiDAR las pendientes locales de cada celda del MDT mediante el algoritmo de Zevenbergen y Thorne el cual obtiene el valor de la pendiente del terreno haciendo uso de un esquema en diferencias finitas de segundo orden con los valores de las cuatro celdas vecinas a la de interés.

  - Estudio hidrológico:



(a) MDT (b) MDS

Figura 2.7: MDT y MDS de un bosque con vegetación intensa de Galicia.

tiene como objetivo conocer los caudales en régimen natural de la máxima crecida ordinaria y de otras avenidas (al menos las de 100 y 500 años) para cada uno de los tramos a estudiar. Esto servirá de entrada para el posterior estudio hidráulico.

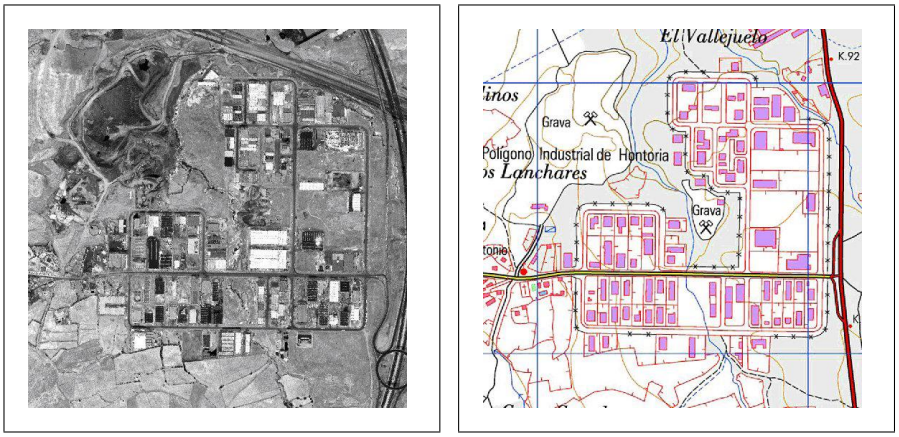
- Estudio hidráulico: conociendo los mecanismos de inundación gracias al estudio geomorfológico y a los hidrogramas y caudales máximos resultantes del estudio hidrológico, los objetivos del estudio hidráulico son para cada periodo de retorno:

  - Obtener los calados máximos alcanzados.
  - Definir la zona de inundación.
  - Si fuera necesario, obtener la distribución de velocidades máximas en la Zona de Inundación.

  En la figura 2.9 se muestra la envolvente de calados máximos obtenidos en la simulación bidimensional de un temporal marítimo en la costa del municipio de Oliva asociado a 500 años de periodo de retorno.

- **Modelos urbanos en 3D.**

- **Generación de mapas de ruido.**



(a) Imagen de intensidades.

(b) Ráster 1-25000.

(c) MDT

(d) MDS

Figura 2.8: Imágenes del proyecto de Segovia realizado por DIELMO 3D S.L.

- **Escenarios virtuales en 3D.**

- **Actualización muy precisa de la línea costera.**

  A partir de productos generados con LiDAR es posible la obtención de la posición de la línea de costa con mucha precisión permitiendo realizar diferentes actuaciones como:

  - Estudios de erosión de costas.
  - Localización precisa de la línea de costa y estudio de su evolución.
  - Control de la morfología de la zona costera en cuanto a su

protección.

- Estudios de inundaciones en deltas y diques.

- **Modelado de líneas de alta tensión.**

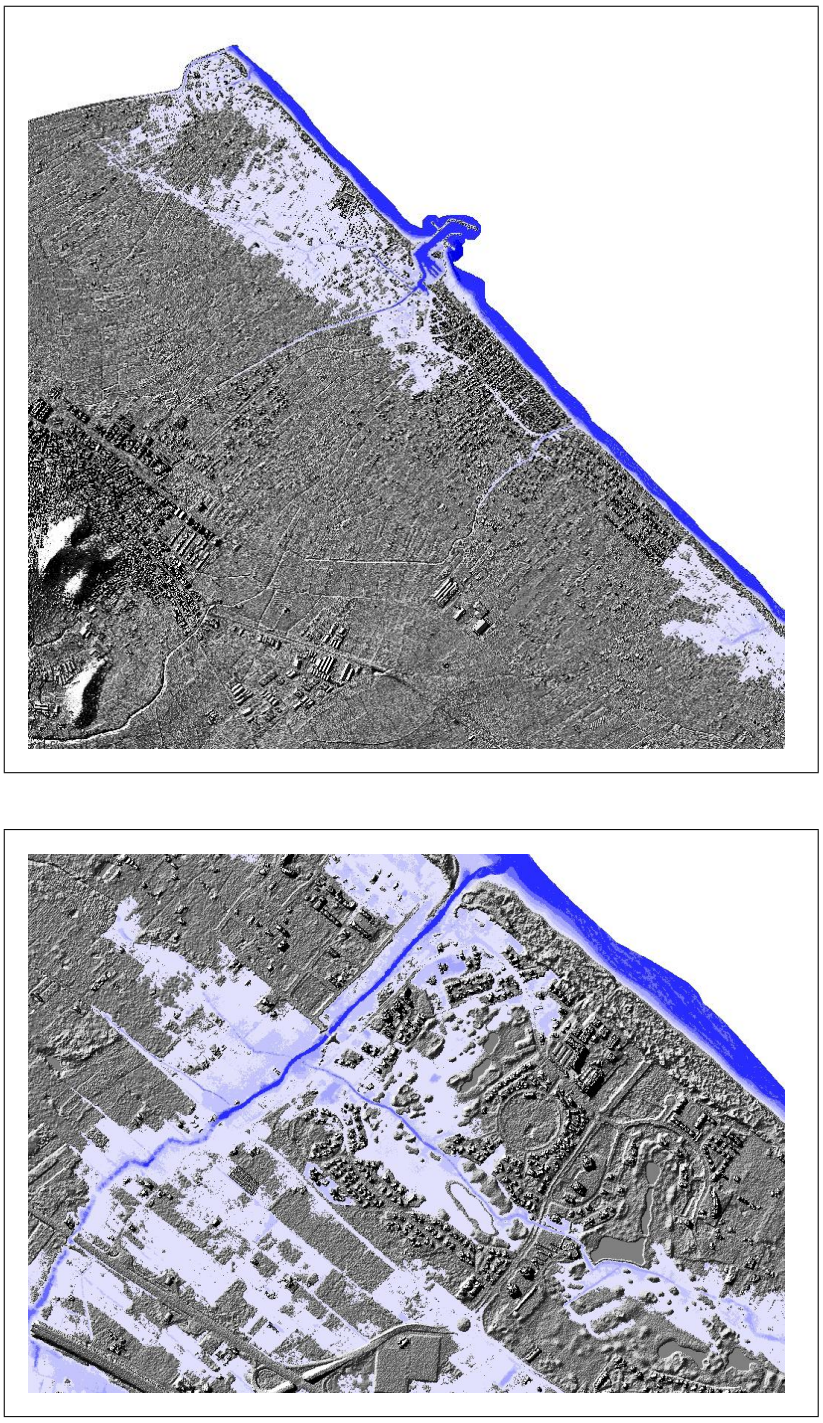


Figura 2.9: Envolvente de calados máximos por inundación costera en el Término municipal de Oliva

CAPÍTULO 3

Antecedentes

3.1 Introducción

En esta sección se van a presentar los conceptos y tecnologías en los que se encuadra la realización de este proyecto. Para de esta forma poder situarse en el contexto adecuado que permita el mejor entendimiento del resto de secciones que componen este documento.

En primer lugar se va a hacer un repaso a los sistemas de información geográfica, se explicarán conceptos básicos sobre su funcionamiento y utilización haciendo énfasis en el sistema de información geográfica gvSIG.

A continuación se analizarán una serie de aplicaciones que permiten la visualización y procesado de datos LiDAR. De esta forma, se tratarán funcionalidades que posteriormente estarán disponibles en las aplicaciones aquí presentadas.

Finalmente, se analizarán otras aplicaciones relacionadas con gvSIG que complementan su funcionalidad. Como es el caso de SEXTANTE

que se trata de una herramienta de análisis geoespacial. Su característica principal es proporcionar una plataforma de desarrollo de geoalgoritmos de manera sencilla.

3.2 Sistema de información geográfica

Un sistema de información geográfica (SIG o GIS con las siglas en inglés) se define como: *Base de datos computerizada que contiene información espacial* [Cebrián y Mark, 1986]. O también como: *Una tecnología informática para gestionar y analizar información espacial.* Otra posible definición más amplia sería: *Un conjunto de herramientas para reunir, introducir, almacenar, recuperar, transformar y cartografiar datos espaciales sobre el mundo real para un conjunto particular de objetos* [Burrough, 1988]. También es importante mencionar la definición que proporciona el National Center for Geographic Information and Analysis de Estados Unidos: *Un sistema de hardware, software, y procedimientos elaborados para facilitar la obtención, gestión, manipulación, análisis, modelado, representación y salida de datos espacialmente referenciados, para resolver problemas complejos de planificación y gestión* [NCGIA, 1990].

Figura 3.1: Componentes de un sistema de información geográfica.

Un Sistema de Información Geográfica se puede contemplar de manera mucho más simple como un conjunto de mapas de la misma porción de territorio, donde un lugar concreto tiene la misma localización en

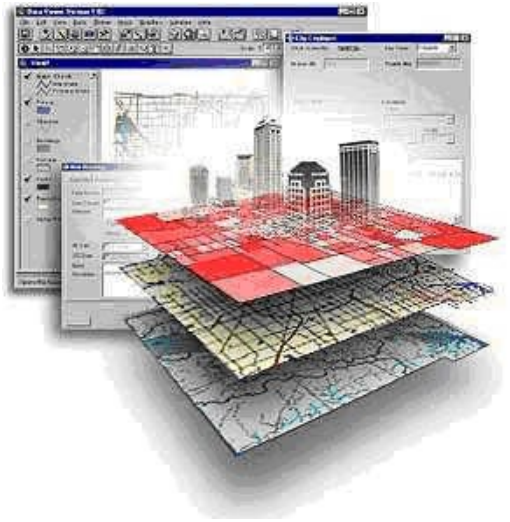


Figura 3.2: Sistema de información geográfica.

todos los mapas incluidos en el sistema de información. De esta manera, resulta posible realizar análisis de sus características geoespaciales y temáticas de manera sencilla para un mejor conocimiento de la zona de estudio. Por lo tanto un sistema de información geográfica puede ser considerado como una tecnología aplicada a la resolución de problemas territoriales, además de ser de gran utilidad en cualquier área y estudio donde el manejo de información espacial sea encesario o de gran valor añadido.

Los sistemas de información geográfica forman parte del ámbito más extenso de los denominados sistemas de la información. Un sistema de la información incluye una base de datos, una base de conocimientos (un conjunto de procedimientos de análisis y manipulación de los datos) y un sistema de interacción con el usuario. De la misma manera estos elementos se pueden encontrar en la organización general de un sistema de información geográfica. Un elemento muy relacionado con los sistemas de la información son los Sistemas de Apoyo a la Decisión, en estos sistemas los datos y la base de conocimientos conforman una estructura para servir de apoyo a la toma de decisiones, de esta forma se facilitan posibles respuestas y simulaciones en caso de optar por una alternativa u otra. Los SIG, en algunos casos, son de manera simultánea, un Sistema de Información y un Sistema de Apoyo a la Decisión.

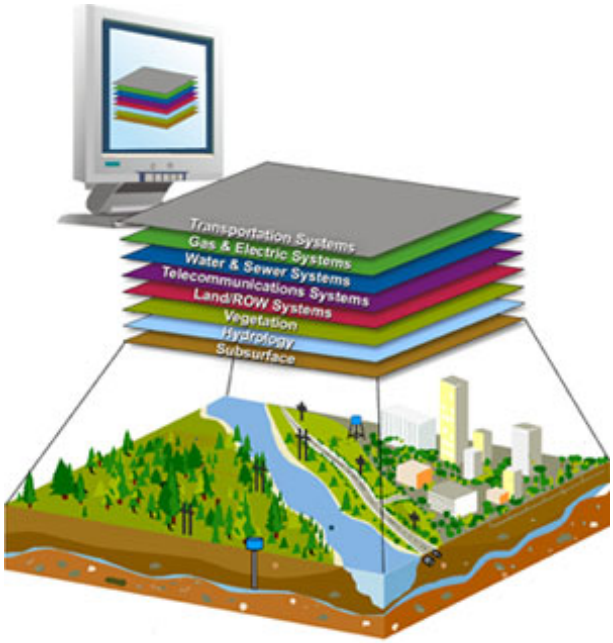


Figura 3.3: Esquema de representación multicapa de un SIG.

En la sociedad de la información es el contexto general donde están surgiendo los sistemas de información geográfica, tal y como se han definido. Esta organización social, producto de una nueva revolución industrial, otorga una gran relevancia a la disponibilidad inmediata, de la información, lo que hace posible resolver problemas y responder preguntas rápidamente. El proceso clave es la comercialización o mercantilización de la información, lo que hace que ésta se convierta en un bien más, que se compra y se vende, como cualquier otro. Las técnicas de análisis geográfico, en este contexto, (existentes en un SIG) *permiten, siendo usadas de modo adecuado, dar un valor añadido a la información y ofrecer excelentes oportunidades para muchas aplicaciones en la vida real* [Openshaw y Goddard, 1987]. Las nuevas tecnologías, como la informática, nuevos sistemas de comunicación, la inteligencia artificial, organización del conocimiento, etc. que dan estructura a la sociedad de la información, son utilizados en los sistemas de información geográfica.

Existe cierta confrontación entre lo que es un sistema de información geográfica y un programa de de cartografía asistida por ordenador o de gestión de base de datos. Hay autores como Dueker [DUEK, 1987] y Cowen [COW, 1988] que insisten en que lo más característico de un

SIG es su capacidad de análisis, de generar nueva información de un conjunto previo de datos mediante su manipulación y reelaboración. Por tanto, un sistema de información geográfica es más que un sistema de diseño asistido por computador (CAD) y lo es por su capacidad de relacionar elementos gráficos (puntos, líneas, polígonos), que también son manejados por los sistemas CAD, con elementos de una base de datos, aspecto del que carecen los sistemas CAD. Por otra parte, los sistemas de información geográfica generan información además de visualizarla a diferencia de los programas de cartografía asistida por ordenador.

Por lo que se puede ver para el manejo de datos espaciales, los sistemas de información geográfica son el paso más importante desde la invención de los mapas.

Los SIG tienen las siguientes **características**:

- Son sistemas diseñados para permitir la visualización de información geográfica expresada en forma de mapas.

- El punto más importante de su funcionamiento se encuentra en la posición de un elemento geográfico representatado por elementos gráficos (puntos, lineas, polígonos) y su información temáticas asociada.

- Sistemas que disponen de un amplio abanico de funciones de análisis y consulta que permiten la explotación de la información geográfica para la resolución de un problema o cubrir una necesidad determinada.

- Son el resultado de aportaciones de otra serie de disciplinas (matemáticas, geografía, cartografía,etc.) de las que han obtenido una capacidad para el manejo de información geográfica.

- Almacenan relaciones espaciales entre los diferentes elementos del sistema. Esto permite interrogar al sistema sobre estas relaciones.

Son por ello sistemas de gran ayuda en la toma de decisiones.

Entre las principales **funciones** que tienen los sistemas de información geográfica destacan las siguientes:

- **Adquisición de la información:** disponen de la funcionalidad necesaria para capturar y depurar errores tanto en la información geográfica espacial como temática.

- **Funciones de gestión:** dotados con la funcionalidad para una estructuración de la información original en distintas capas de información ordenada y coherente.

- **Funciones de almacenamiento:** es importante que los sistemas SIG puedan hacer uso de sistemas de gestión de base de datos (SGBD). Su uso facilita las tareas de actualización y manipulación de los datos.

- **Funciones de análisis:** son las que aportan al sistema de información geográfica su mayor potencialidad. Permiten la obtención de información no presente a simple vista, generando nuevos datos y realizando simulaciones de comportamientos que se basan en modelos territoriales.

- **Funciones de salida:** permiten mostrar finalmente tanto los datos del propio sistema como los resultantes tras aplicar funciones de análisis sobre ellos.

Habitialmente al trabajar con sistemas SIG el mayor coste en cuanto a tiempo y dinero suele darse en la captura y mantenimiento de la información geográfica. Su constante actualización es imprescindible. Los datos son considerados como algo vivo y deben ser actualizados tanto los geográficos como los temáticos para evitar que el paso del tiempo influya en la calidad de los mismos. Por ello es necesaria una

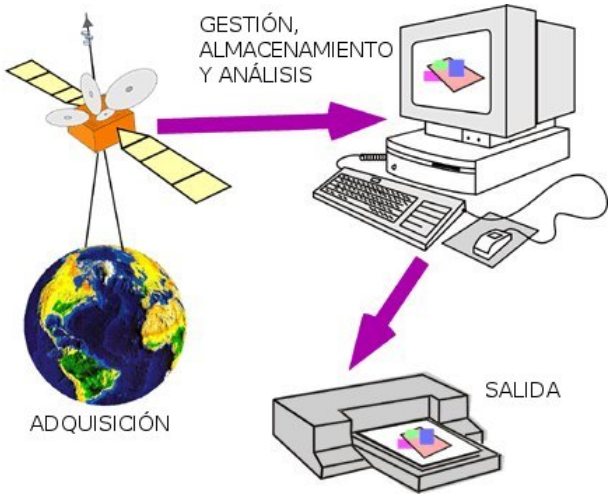


Figura 3.4: Esquema de las funciones GIS.

metodología de captura de información y actualización para mantenerla al día para de esta manera asegurar una calidad óptima en los análisis realizados en el sistema a partir de dicha información.

3.2.1 Aplicaciones de los SIG.

Los SIG tienen aplicación a casi cualquier actividad humana. Son aptos para la resolución de cualquier problema que dependa de una variable que esté asociada a un elemento geográfico.

El aumento de la demanda de Sistemas de Información Geográfica es debido a su versatilidad y a su capacidad de adaptación para aportar soluciones empresariales, a un amplio abanico de sectores, pese a que algunos de ellos no pertenezcan, a priori, al grupo de clientes potenciales de un sistema de información geográfica. Diferentes sectores explotan las capacidades de los SIG para **aplicaciones** de distintas naturalezas como son:

- **Logística:** En la gestión del tráfico terrestre aéreo o marítimo el análisis de rutas óptimas para el reparto y distribución de mercancías. Los sistemas de gestión de flotas en compañías de transportes o flotas de taxis son cada día más comunes para la optimización de los recursos de la empresa. Son bien conocidas las funciones de cálculo de la ruta más corta, la más económica y la más rápida en los dispositivos GPS para vehículos.

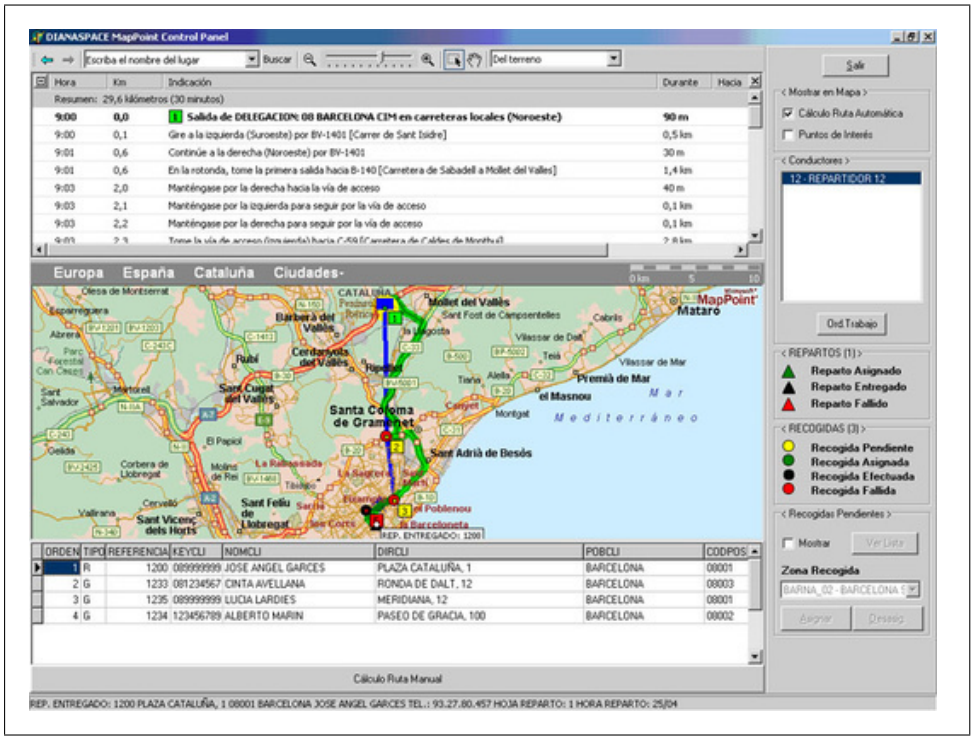


Figura 3.5: Interfaz de un sistema de gestión de flotas.

- **Planificación urbanística, territorial:** El urbanismo que es la disciplina que tiene como objetivo de estudio a las ciudades, desde una perspectiva holística enfrenta la responsabilidad de estudiar y ordenar los sistemas urbanos. También es la forma en que los edificios y otras estructuras de las poblaciones se organizan o la agregación y forma de estar distribuidas las poblaciones en núcleos mayores como ciudades. Las aplicaciones SIG dan un soporte integral a este grupo.

- **Control catastral y registral:** El Catastro ha de conciliar la parcela catastral con la realidad inmobiliaria. El Registro la descripción literaria de la finca con su base gráfica registral. En la actualidad el Colegio de Registradores de la Propiedad en España utiliza una aplicación llamada GEOBASE. Se trata de una aplicación SIG que permite la digitalización de bases gráficas registrales por parte de técnicos SIG y su validación posterior por parte del registrador de la propiedad correspondiente.

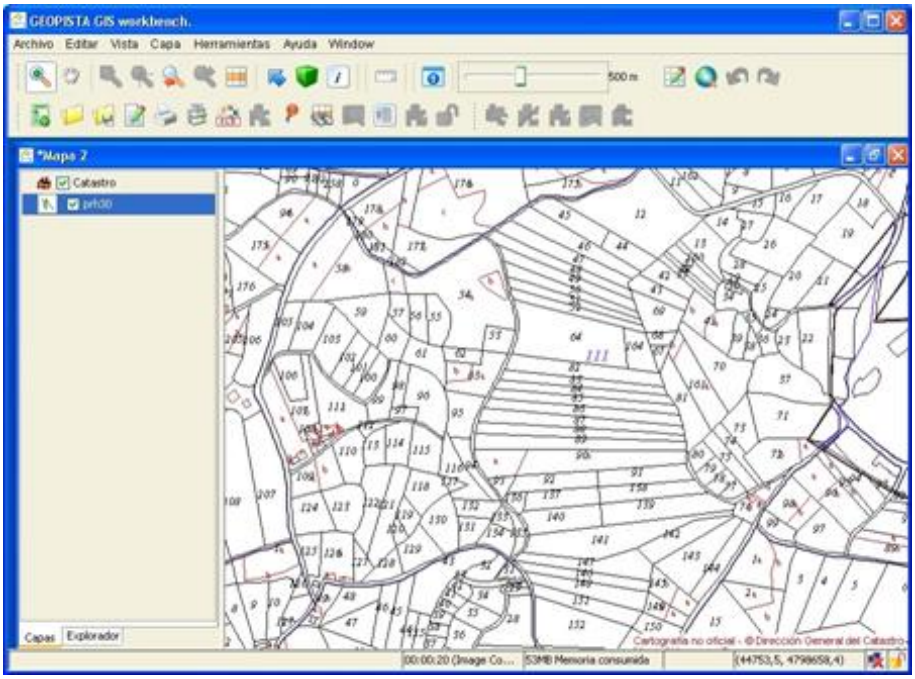


Figura 3.6: SIG con una capa vectorial visible del catastro.

- **Administración pública:** Cada vez más las administraciones locales hacen uso de las herramientas SIG para un correcto funcionamiento de sus tareas administrativas. Como un ejemplo de este grupo que utilizan un sistema basado en gvSIG merece especial mención la EIEL[1] de la diputación de Pontevedra. El objetivo inicial de EIEL consiste en el análisis y valoración de las necesidades de las entidades locales en cuanto a dotaciones de infraestructuras y equipamientos se refiere. Como parte de su apuesta por la mejora y la modernización de sus procesos de planificación y gestión de las infraestructuras territoriales, las Administraciones Públicas están apostando por el uso de las Tecnologías de Información Geográfica y de la Comunicación y la EIEL es un ejemplo de esta afirmación. En este contexto, la Diputación de Pontevedra ha apostado por la implantación de un SIG corporativo y libre basado principalmente en gvSIG y en la adaptación de herramientas de gisEIEL. La iniciativa no sólo mejora el trabajo de los técnicos provinciales, sino que también permite presentar un conjunto de extensiones para gvSIG de alto interés para la comunidad de usuarios de SIG.

[1]Encuesta de Infraestructura y Equipamientos Locales. Inventario periódico demandado por el Ministerio de Política Territorial español a las diputaciones provinciales.

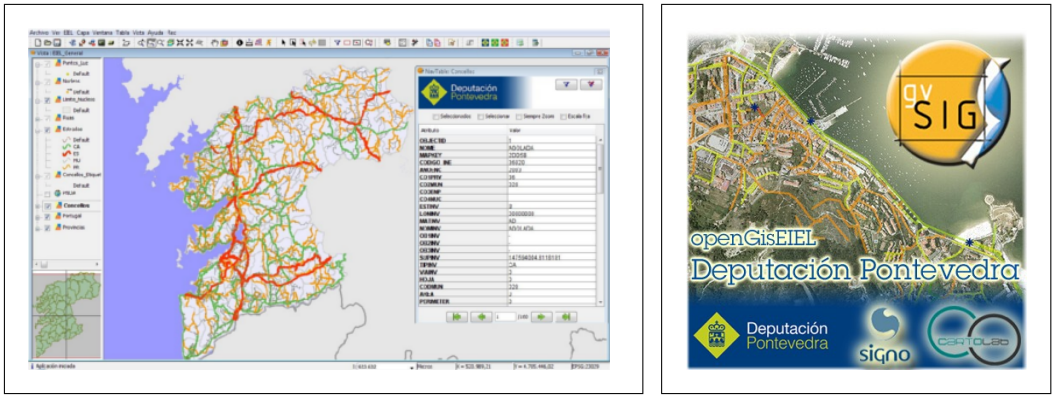


Figura 3.7: Sistema basado en gvSIG gisEIEL.

Figura 3.8: Aplicación GIS para la dministración local. Diputación de Jaén.

- **Explotación de recursos:** Sistemas de información geográfica destinados a la evalución de zonas de yacimientos minerales, petrróleo, gas, etc.

- **Análisis de mercados o geomárketing:** A la hora de realizar un estudio para la creación de comercios es muy importante la realización de un análisis poblacional y demográfico. La aplicación de geoprocesos como el cálculo del área de influencia unido a más

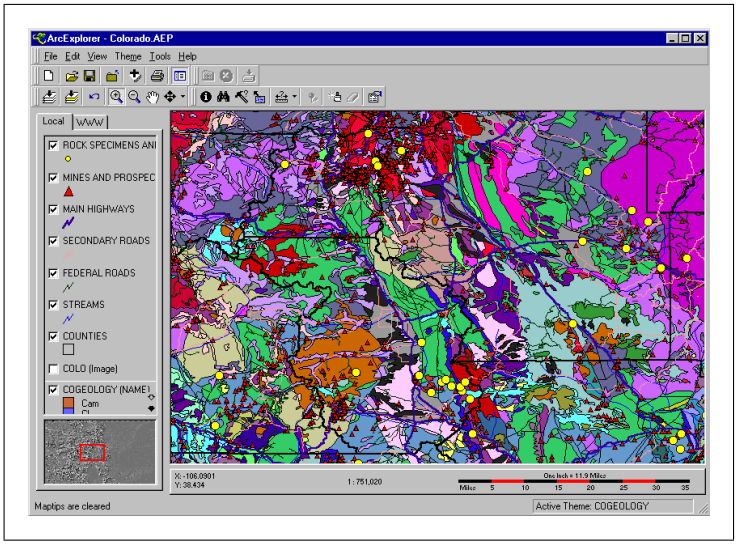


Figura 3.9: Aplicación GIS para la explotación de racursos naturales.

parámetros que intervengan en el problema en cuestión pueden resultar necesarios en la toma de decisiones para ubicación de nuevos negocios. Segmentaciones de mercado, distribución territorial de la población y de las características socioeconómicas.

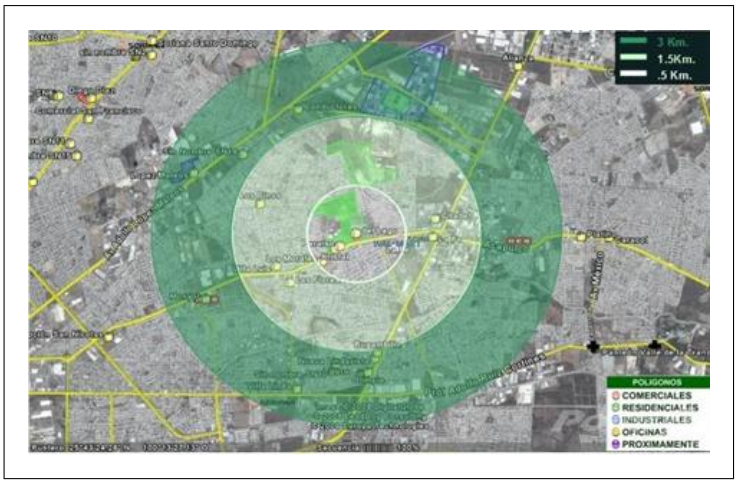


Figura 3.10: Aplicación GIS mostrando cálculos de áreas de influencia.

- **Turismo:** No es extraño que la localización de puntos de interés turísticos haya sido uno de los campos de investigación pioneros puesto que, en este caso, se requiere que el lugar o zona seleccionado tenga unas determinadas características que los clientes valoran. De esta forma las promociones turísticas de pueblos, municipios, comunidades y paises incorporan cada vez más a menudo

tecnología SIG para tal finalidad. Representación de rutas turísticas, puntos de interés, aplicaciones web SIG que permiten a los usuarios participar en la actualización de información geográfica y temática. Por ejemplo el IGN[2] ha publicado mediante un Servicio Web de Mapas WMS una colección de mapas a escala 1:50.000 que cartografían el Camino de Santiago siguiendo el itinerario más popular, con información temática que será de utilidad para los peregrinos que durante el año 2010, Año Xacobeo, decidan recorrer el Camino. La dirección del servicio es http://www.idee.es/wms/IGN-Camino-Santiago/IGN-Camino-Santiago y las capas disponibles son ciudades del camino, ciudades de parada, etapas y un mapa general en formato raster. En la imagen 3.11 se observa del servicio cargado en Kosmo.

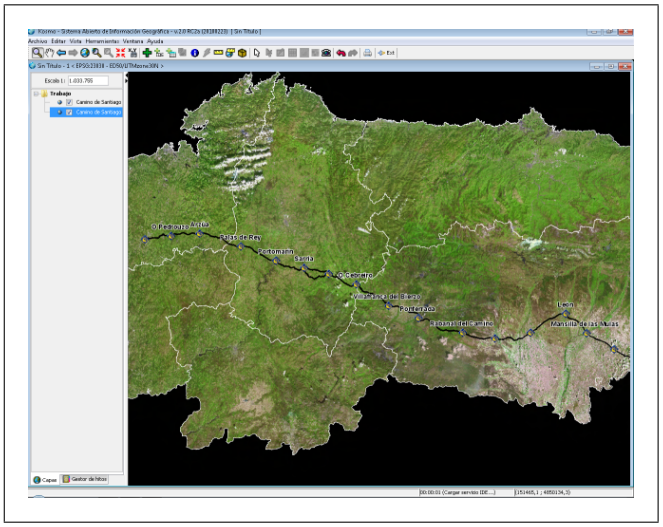


Figura 3.11: Servicio WMS del Camino de Santiago.

- **Salud pública:** Aplicaciones para la gestión e intervención sanitaria en emergencias como el envio de ambulancias. Control y estudio de pandemias. En la imagen 3.12 se muestra una aplicación SIG que realiza un estudio sobre la distribución geográfica de casos de cáncer diagnosticados durante un periodo determinado de tiempo.

- **Seguridad pública:** Aplicaciones SIG por la policía para el control de la criminalidad (ver figura 3.14). En las cuartas jornadas

[2]Instituto Geográfico Nacional

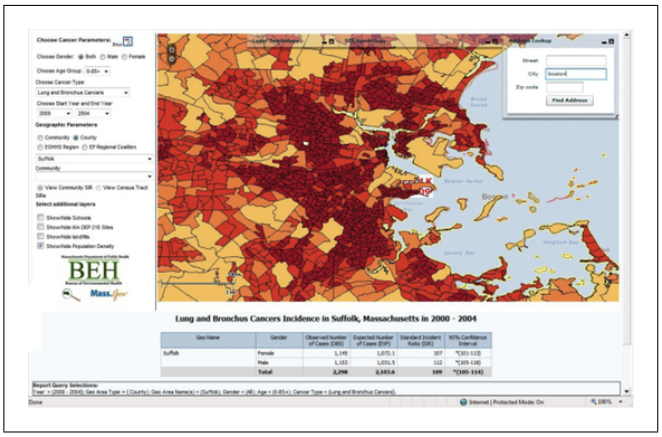


Figura 3.12: Aplicación GIS para el estudio de la salud pública.

de gvSIG se presentó una aplicación basada en gvSIG (ver figura 3.13) que responde a las zonas de planificación, según los planes de emergencia nuclear del nivel de de respuesta exterior. Para situaciones de grave riesgo colectivo, calamidad pública o catástrofe extraordinaria, en las que la seguridad y la vida de las personas pueden peligrar y sucumbir masivamente, generándose unas necesidades y recursos que pueden exigir la contribución de todas las Administraciones públicas, organizaciones, empresas e incluso particulares.

- **Aplicaciones medioambientales:** En la prevención y gestión de catástrofes naturales como incendios, inundaciones, terremotos, corrimientos de tierra, inventarios de suelos o controlar el tipo del uso del mismo etc. En las quintas jornadas de gvSIG se presentó una aplicación basada en gvSIG cuya finalidad era el estudio relativo al monitoreo de incendios (ver imagen 3.15). Esta aplicación tiene como objetivo comparar la renovación de la vegetación de bosque de Karst y bosques de pinos quemados en áreas con la evolución natural y misma vegetación que en las zonas no dañadas por el fuego que se produjo en julio de 2003. Para esta comparación se ha efectuado el cálculo de los años necesarios para los índices NDVI [3] y NDWI [4] , obtenidos a partir de imágenes multiespectrales del satélite Landsat, para alcanzar los valores encontrados en áreas no expuestas al fuego. El NDVI cuantifica la

[3]Normalized Difference Vegetation Index o Índice de Diferencia de Vegetación Normalizado
[4]Normalized Difference Water Index o Índice de Diferencia Normalizada de agua

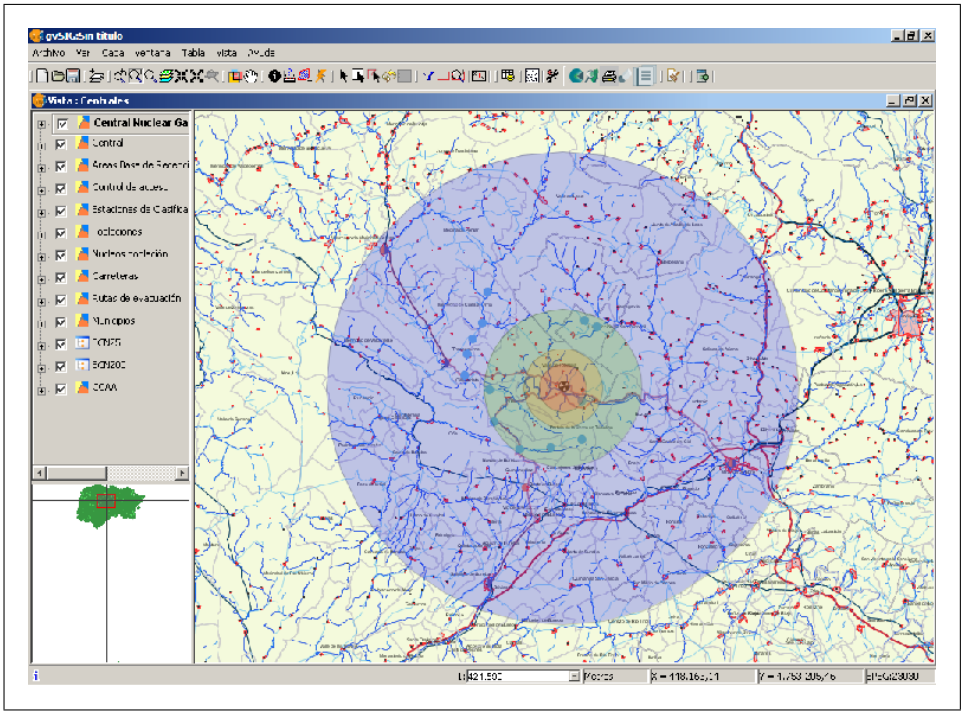


Figura 3.13: Aplicación GIS para situaciones de catástrofe.

biomasa verde, mientras que el NDWI se ve afectada por contenido de agua de la hoja y la humedad del suelo. Aunque la estructura de la vegetación en las áreas quemadas se encuentra todavía en una etapa de evolución, después de 5 años, los dos índices casi coinciden con los valores área inalterada. La empresa DIELMO 3D S.L. ha eralizado estudios de inundabilidad de municipios como Oliva (ver imagen 2.9).

- **Educación:** Los métodos de enseñanza evolucionan y se integran con lo que ha dado por llamarse sociedad de la información. Un ejemplo de aplicación SIG para la educación lo encontramos en EduSIG (ver figura 3.17). Se trata de una aplicación basada en gvSIG aplicada a la enseñanza de la geografía. El mecanismo de enseñanza se realiza mediante juegos educativos basados en mapas. Juegos distintos como: juegos de nombres, de banderas, de contornos, localización de topónimos con un sistema de puntuaciones y configurables.

- **Aplicaciones militares:** permiten almacenar, analizar y actualizar información militar geográfica necesaria para la toma de

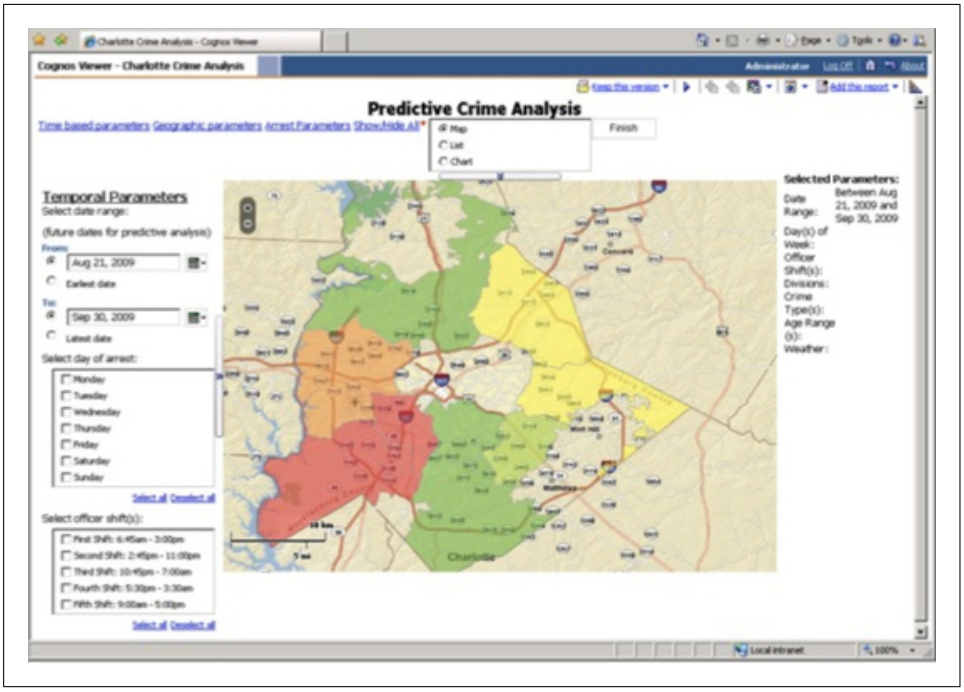
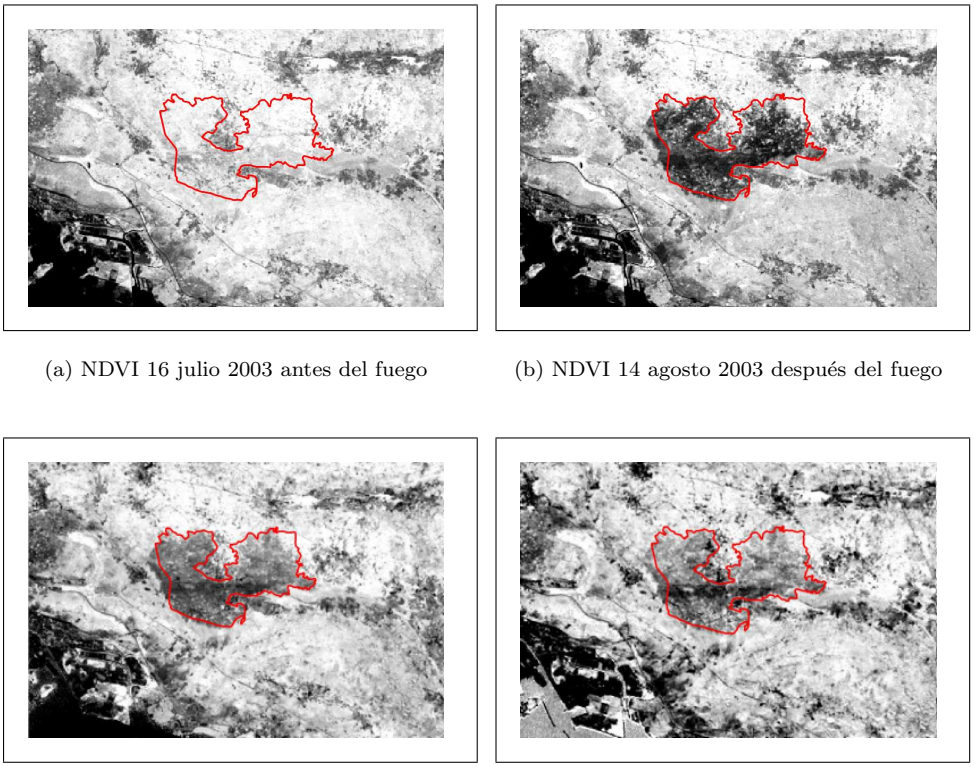


Figura 3.14: Aplicación GIS para el estudio de la criminalidad.

decisiones y posterior resolución del comandante y sus asesores en el nivel estratégico, operativo y táctico. Un SIG. militar debe servir a la conducción estratégica, operativa y táctica, permitiendo integrarse a ellos: análisis de cambios temporales en los escenarios geográficos, establecimiento de bases de datos de objetivos militares, actualización y análisis de posible zonas de objetivos, análisis en tres dimensiones de Líneas de Operaciones, localización de Objetivos, proporcionar información y evaluar conforme a parámetros establecidos, posibles sitios para la detección y obtención de agua, despliegue de instalaciones logísticas y administrativas, posibles ubicaciones para el despliegue de armas pesadas para la infantería y Artillería, etc., integración a sistemas de mando y control para el apoyo a la toma de decisiones en el campo táctico, operativo y estratégico.

- **Redes sociales:** Una red social es una estructura social compuesta de personas, organizaciones u otras entidades, las cuales conectadas por uno o varios tipos de relaciones, como amistad, parentesco, intereses comunes, intercambios económicos, relaciones sexuales, o que comparten creencias, conocimiento o prestigio. Son



(a) NDVI 16 julio 2003 antes del fuego

(b) NDVI 14 agosto 2003 después del fuego

(c) NDVI 29 julio 2005

(d) NDVI 24 julio 2009

Figura 3.15: Aplicación de gvSIG en un estudio relativo al monitoreo de incendios. (Quintas jornadas gvSIG)

muy conocidas redes sociales como Facebook, Twitter, Linkedin, Flickr, Tuenti, etc. Aunque no es tan habitual las redes sociales pueden incorporar aplicaciones SIG para compartir información temática asociada a una posición geográfica determinada. La aplicación de Nokia Sports Tracker aplicada a fitness permite los datos sobre sus viajes a través de un mapa de viaje acompañado por imágenes y videos tomados en el camino.

- **telecomunicaciones:**planificación de las redes de telefonía móvil, de televisión por cable, análisis de cobertura del medio. Por ejemplo SIGNET es un sistema de información para la gestión local de redes externas, desarrollado con el apoyo de herramientas SIG, para garantizar la consistencia e integridad de la conectividad entre los elementos que componen la red local. Permite la incorporación y actualización de redes, asignación, administración de redes, mantenimiento y administración cartográfica.

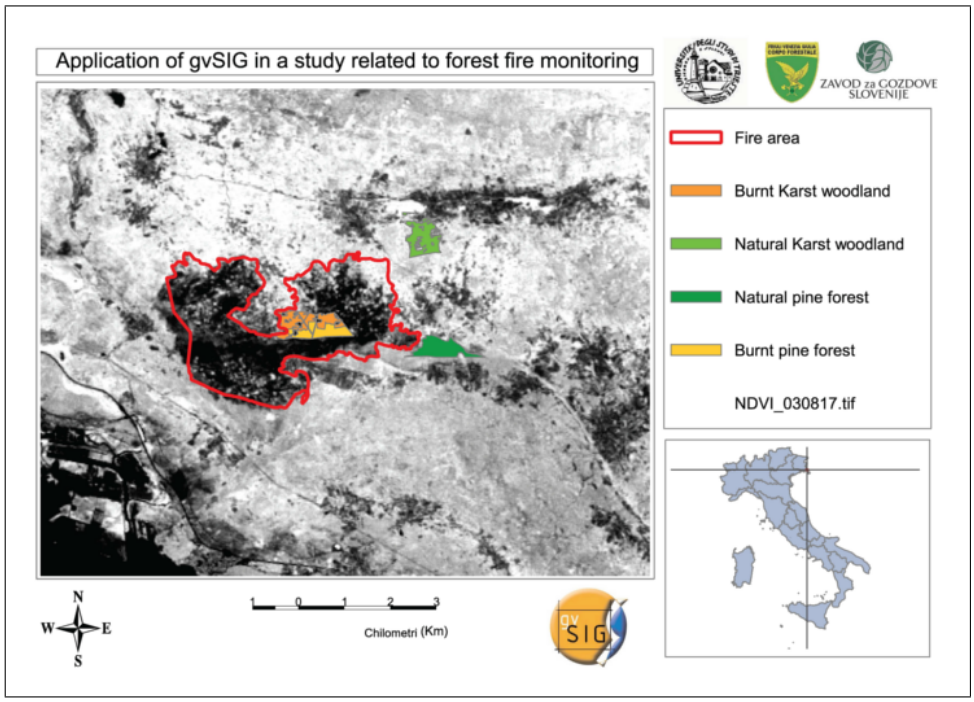


Figura 3.16: Salida gráfica de la aplicación de estudio relativo al monitoreo de incendios.

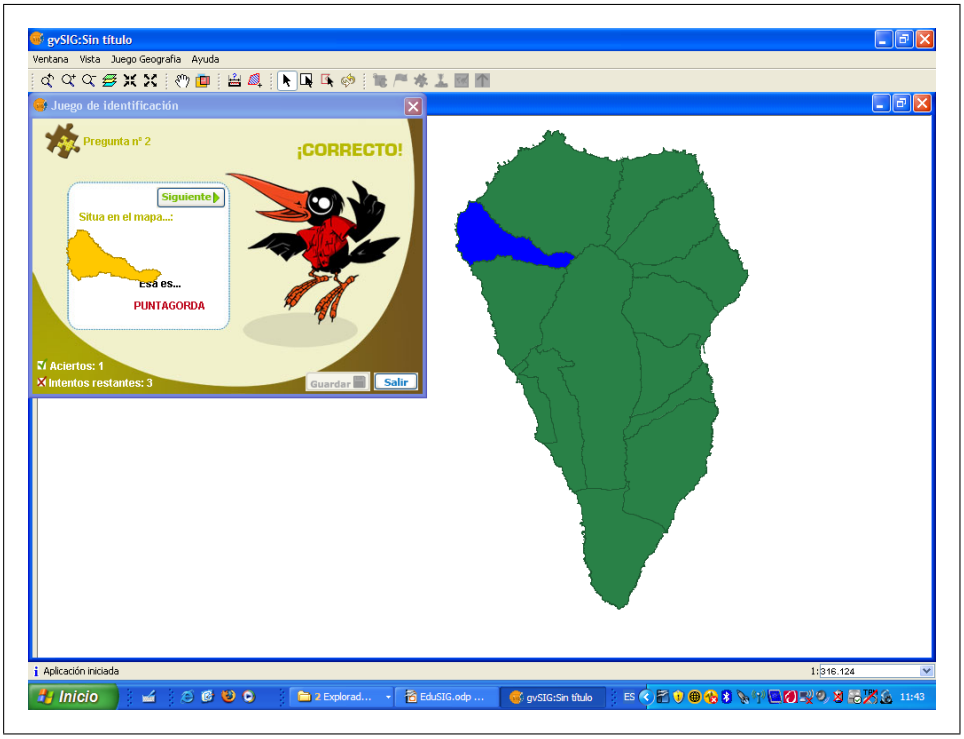


Figura 3.17: Aplicación GIS parala enseñanza de geografía.

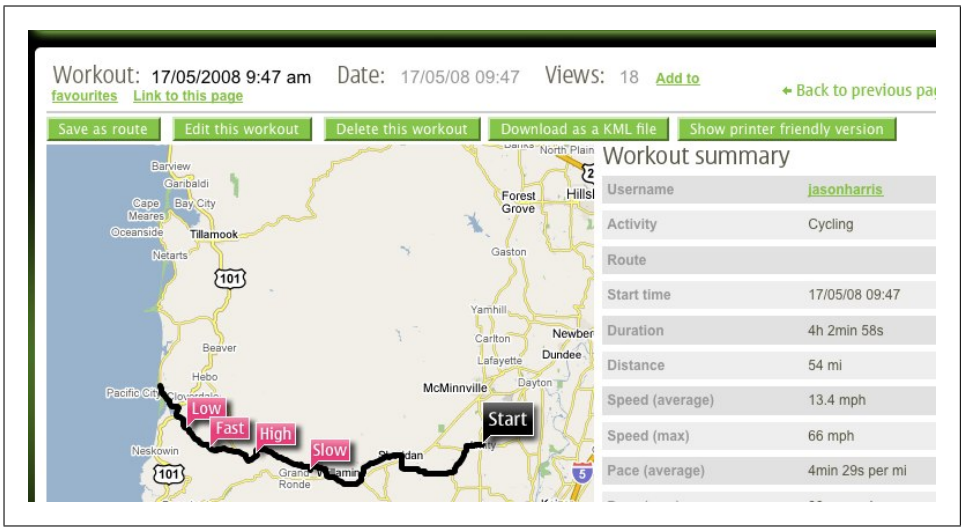


Figura 3.18: Nokia sport tracker. GPS tracker aplicado a prácticas deportivas.

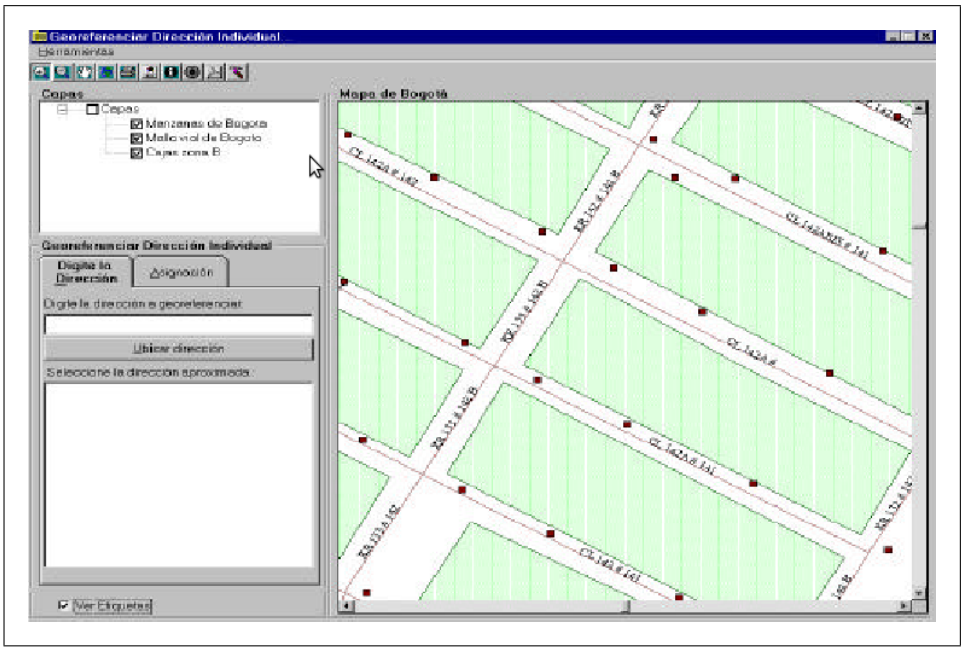


Figura 3.19: Aplicación Asignet para la gestión de redes de fibra óptica.

La demanda de estos sistemas de información ha propiciado la aparición de un gran número de estas aplicaciones, tanto propietarias como con licencia GPL. Destacan, entre los de licencia no pública ArcGis de ESRI o MapInfo y de libre distribución gvSIG.

3.2.2 **Modelos de representación.**

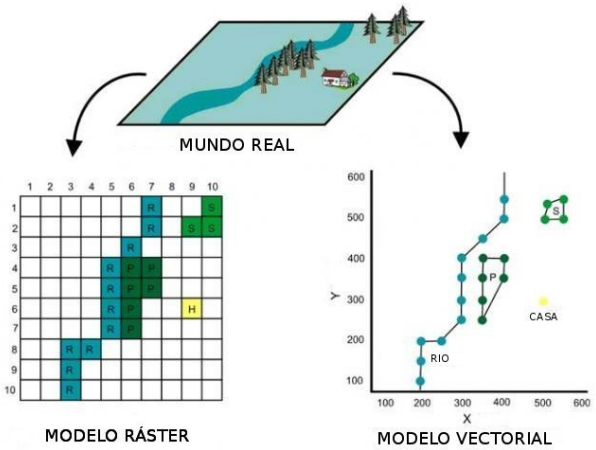


Figura 3.20: Esquema de los modelos ráster y vectorial.

Los sistemas de información geográfica han de ser capaces de representar y almacenar las entidades geográficas reales mediante entidades gráficas.

El mapa tradicional es una representación analógica (continua) de la realidad, por tanto, no está preparado para ser procesado por un computador que hace uso de los datos digitales (discreto). Por eso en primer lugar es necesario realizar una conversión al formato digital para introducir los datos en un sistema de información geográfica. El principal problema de este proceso es la representación digital de la componente espacial de los datos geográficos.

Para una correcta representación digital de los datos espaciales es necesaria la resolución de dos cuestiones: la geocodificación de los datos y la descripción de las características espaciales en términos digitales. La primera consiste en un procedimiento por el cual un objeto geográfico (un edificio, una parcela, una carretera, etc.) recibe directa o indirectamente una etiqueta identificando así su posición espacial con respecto a algún marco de referencia o punto común.

El proceso de geocodificación, que determina la localización de cada objeto geográfico, se puede llevar a cabo de dos formas diferentes. Directamente, usando un sistema de ejes de coordenadas respecto a los que se determina la posición absoluta de cada punto; e indirectamente, asignando a cada objeto una referencia espacial que lo distingue de los restantes y permite establecer su posicionamiento relativo respecto a los demás, las direcciones postales son un buen ejemplo de geocodificación indirecta.

En un segundo lugar, se debe realizar una descripción de la posición geométrica de cada objeto y de las relaciones espaciales, la topología, que mantiene con los restantes objetos geográficos existentes en la realidad a estudiar. Para realizar esta última labor es imprescindible la abstracción y simplificación de los elementos existentes, es decir, la creación de un modelo de datos con todos los elementos representados digitalmente. Un modelo de datos es, según la informática y la teoría de bases de dato, un conjunto de directrices que permiten la representación lógica de los datos en una base de datos, consistente en los nombres de las unidades lógicas de los datos y de las relaciones existentes entre ellos. Un modelo es siempre, una representación simplificada de la realidad, o como dice Peuquet, *una abstracción del mundo real que incorpora sólo aquellas propiedades que son relevantes a la aplicación de interés en cada caso*. De esta manera existen varios tipos de modelos de datos de los objetos geográficos entre los que destacan: el modelo vectorial y el modelo raster. La principal diferencia entre ambos es el modo de guadar la información.

En el caso del modelo ráster se almacena una matriz de posiciones. Cada posición de esta matriz representa una fracción del objeto que representan y toma el valor del objeto real en ese punto. Por el contrario el modelo vectorial almacena las coordenadas de la geometríá que representan.

Podemos encontrar diferentes ejemplos de utilización de estos modelos. En aplicaciones de retoque fotográfico se emplea el modelo ráster. En aplicaciones CAD se utiliza el modelo vectorial.

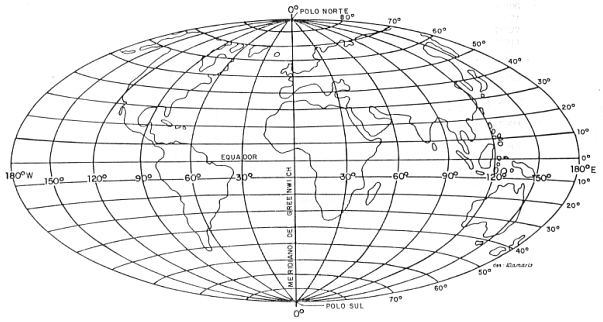


Figura 3.21: Sistema de coordenadas geográficas que utiliza las dos coordenadas angulares latitud (norte o sur) y longitud (este u oeste) para determinar las posiciones de la superficie terrestre.

Dependiendo del contexto en el que nos situemos será conveniente la utilización de un modelo frente a la del otro. Las principales **ventajas del modelo vectorial** son:

- Ocupa menos espacio de almacenamiento.
- Tiene una alta precisión en la definición de entidades geométricas.
- Representa adecuadamente las relaciones topológicas.
- Es posible obtener mejores salidas gráficas.

Los principales **inconvenientes del modelo vectorial** son:

- La captura de los datos es más costosa.
- La estructura de datos es más compleja.
- Existe una mayor dificultad cuando se realizan ciertas operaciones como es el caso de comparaciones de mapas.

Las principales **ventajas del modelo ráster** son:

- La captura es más sencilla que en el modelo vectorial.
- La estructura de datos es más simple.
- La manipulación y gestión de la información es más sencilla.

Los principales **inconvenientes del modelo ráster** son:

- Tiene menor precisión en ciertos cálculos geométricos como son el cálculo de áreas y longitudes.
- Ocupa un mayor espacio de almacenamiento que el modelo vectorial.
- Mayor dificultad en la representación de ciertas relaciones topológicas.

Por lo que se ha enumerado se recomienda de forma general trabajar con el modelo de información ráster en las siguientes ocasiones:

- Cuando se trabaje con grandes extensiones de terreno y a escalas pequeñas.
- Cuando no se requiera mucha precision en los cálculos a realizar.
- Cuando se requiera un análisis rápido en estudios de variabilidad temporal.

Se pueden dar las condiciones mencionadas en casos de estudio sobre impacto medioambiental, recursos naturales, estudios globales de climatología, estudios regionales, cálculo de superficie que se han visto afectada por un incendio.

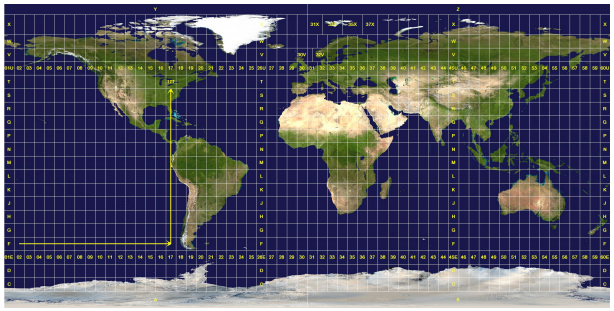


Figura 3.22: Sistema de coordenadas UTM (Universal Transverse Mercator).

Por otro lado se recomienda la utilización del formato vectorial en las siguientes ocasiones:

- Cuando se trabaje con grandes extensiones de terreno pequeñas y medianas y a grandes escalas.

- Cuando se requiera mucha precision en los cálculos a realizar y en la definición de entidades.

- Cuando se necesiten representar relaciones topológicas complejas.

Es posible encontrar estos criterios en casos como la planificación urbanística, la gestión catastral, estudios hidrográficos, arqueológicos, etc.

El modelo vectorial representa objetos espaciales codificando explícitamente sus fronteras. Las líneas que actúan de fronteras se representan mediante las coordenadas de los puntos o vértices que delimitan los segmentos que las forman. Es un modelo semejante a la percepción humana del espacio. Los elementos de referencia de esta representación son los puntos, las líneas y los polígonos destacando como elemento fundamental el segmento lineal. A continuación se detallan los elementos de referencia:

- **Coordenada**: Consiste en un par de números que indican las distancias horizontales con los ejes de coordenadas. Se utiliza para representar localizaciones en la superficie de la tierra.

- **Punto**: Abstracción de un objeto de cero dimensiones. Se representa por un par de coordenadas X,Y. Normalmente un punto representa una entidad geográfica pequeña como para ser representada con una línea o una superficie

- **Línea o arco**: Conjunto de pares de coordenadas ordenados que representan la forma de entidades geográficas que son muy finas como para ser representadas con una superficie (carreteras, curvas de nivel, calles, líneas ferroviarias o ríos), o entidades lineales sin área como por ejemplo los límites administrativos.

- **Polígono**: Entidad utilizada para la representación de superficies. Un polígono está definido por las líneas que forman su contorno y por un punto interno que lo identifica.

Los SIG pueden llevar a cabo una tranformación de los datos en diferentes formatos. Es posible convertir una imagen satélite a un mapa de elementos vectoriales mediante la vectorización o generación de líneas en torno a celdas con una misma clasificación determinando la relación espacial de estas, tales como proximidad o inclusión.

Mediante algoritmos avanzados es posible la vectorización no asistida de imágenes raster. Se trata de una técnica desarrollada desde finales de los años 60 del siglo XX. Para poder realizarla se recurre a ciertas técnicas como la mejora del contraste, imágenes en falso color y el diseño de filtros que implementan transformadas de Fourier en dos dimensiones.

Al proceso de conversión inverso de datos vectorial a una estructura de datos basada en un matriz raster se le conoce como rasterización.

Dado que los datos digitales se capturan y se almacenan tanto en forma vectorial como en raster, un sistema de información geográfica debe ser capaz de convertir de una estructura de almacenamiento a otra los datos geográficos.

3.2.3 **Flujo de trabajo en un sistema de información geográfica.**

Para la correcta comprensión y obtención del máximo rendimiento de un sistema de información geográfica es imprescindible el conocimiento de las fases que se han de aplicar para alcanzar la solución del problema planteado. Las fases son las siguientes:

- **Adquisición de la información:** en todo sistema de información la materia prima son los datos. Estos se estructuran para conformar la información. La calidad de los datos influirá en la calidad de la solución. Hay ditintas formas para capturar la información espacial: escáner, tabletas digitalizadora, levantamiento topográfico o fotogramétrico, plataformas de satélite, GPS tracking [5] , etc. De la misma manera los datos temáticos serán adquiridos de bases de datos, fichas, encuestas, etc.

- **Preparación de la información:** la información recogida en la fase anterior debe ser tratada previamente para depurarla y eliminar errores en su adquisición. Por otra parte debe ser estructurada de forma que su consulta y análisis por parte del sistema sea lo más eficiente posible. En el caso de la información en formato ráster será necesario proceder a su georeferenciación y corrección de posibles deformaciones y errores derivados de su adquisición. En el caso del formato vectorial ciertos procesos como cierre de polígonos, lineas interconectadas correctamente, etc pueden ser necesarios antes de proceder al análisis de los datos. De esta forma es posible estructurar la información guardando las relaciones topológicas entre las entidades representadas.

- **Unión de la información temática y espacial:** en esta fase se le asocia a los elementos geográficos información externa de distinta naturaleza. Los elementos geográficos pasan a estar enlazados de manera unívoca con la información asociada de manera que es posible obtener dicha información a partir del elemento geográfico

[5] dispositivo gps móvil que muestrea su localización a lo largo del tiempo según una frecuencia de muestreo dada que dependa de la velocidad del objeto que lo lleva instalado. Utilizado en sistemas de control de flotas de vehiculos, estudio y mejora de rendimiento deportivo, posicionamiento de aves, etc.

(a) Estación total

(b) GPS

(c) Tableta digitalizadora

(d) Satélite

(e) GPS tracker

(f) Escáner láser 3D

Figura 3.23: Dispositivos para la adquisición de datos espaciales.

y viceversa.

- **Análisis de la información:** Una vez se dispone de los datos unidos puede procederse al análisis de los mismos atendiendo a los criterios de resolución del problema caso de estudio. El análisis puede ser de diversas naturalezas pudiendo ser puramente análisis espacial o tratarse de un análisis híbrido tratando de manera conjunta tanto la información espacial como la de otro origen.

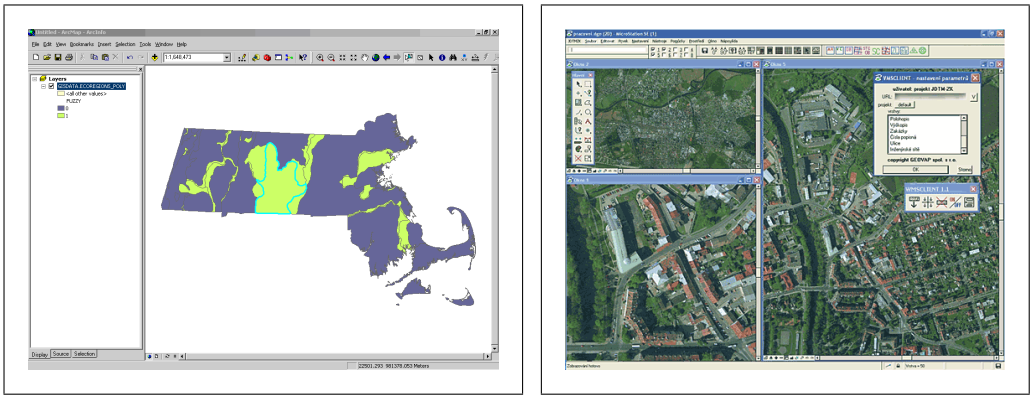
3.2.4 Infraestructuras de Datos Espaciales

Figura 3.24: Logo del OGC.

En los últimos años, el interés en la utilización de la información geográfica para el desarrollo ha llevado a multitud de países y organizaciones a adoptar una combinación de mecanismos, técnicas y políticas encaminados a poder compartir la información espacial a través de sus distintos grupos de trabajo. Estos mecanismos se conocen como Infraestructuras de Datos Espaciales (IDE). Las IDE incentivan la capacidad de los países, los gobiernos locales y regionales y las organizaciones para que compartan conocimientos e información espacial. El Open Geospatial Consortium (OGC http://www.opengeospatial.org) es un organismo impulsor de los metadatos, lenguajes y formatos utilizados para el intercambio de datos, y el proyecto INSPIRE (Infrastructure for Spatial Information in Europe) propone normativas de carácter comunitario que tengan relación con la información geográfica para la Unión Europea. La iniciativa INSPIRE y la IDE de carácter público plantean la evolución constante de los Sistemas de Información Geográfica a clientes de IDE. A continuación se describen los estándares del OGC.

3.2.4.1 WFS (Web Feature Service):

WFS es un servicio estándar, que ofrece una interfaz de comunicación para interactuar con los mapas servidos por el estándar WMS, como por ejemplo, editar la imagen que nos ofrece el servicio WMS o analizar la imagen siguiendo criterios geográficos. Para realizar es-



(a) Ejemplo de capa WFS en un SIG.

(b) Ejemplo de capa WMS en un cliente SIG.

Figura 3.25: Infraestructuras de datos espaciales.

tas operaciones se utiliza el lenguaje GML que deriva del XML, que es el estándar a través del que se transmiten la ordenes WFS. WFS no transaccional permite realizar consultas y recuperar elementos geográficos. Por contra WFS-T (Web Feature Service Transactional) permite además la creación, eliminación y actualización de estos elementos geográficos del mapa.

3.2.4.2 WMS (Web Feature Service):

WMS es un estándar que produce mapas de datos georeferenciados, de forma dinámica. Este estándar internacional define mapa como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador. Los mapas producidos por un servicio WMS se generan generalmente en un formato de imagen (PNG, GIF o JPEG) y opcionalmente como gráficos vectoriales en formato SVG (Scalable Vector Graphics) o WebCGM (Web Computer Graphics Metafile). El estándar WMS define tres operaciones: Devolver metadatos del nivel de servicio, devolver un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos y devolver información de características particulares mostradas en el mapa (opcionales). Las operaciones WMS pueden ser invocadas usando un navegador estándar realizando peticiones en la forma de URLs (Uniform Resource Locators)..El servicio permite así la creación de una red de servidores distribuidos de mapas, a partir de los que los clientes pueden construir mapas a me-

dida. Operaciones WMS también pueden ser invocadas desde clientes avanzados como sistemas de información geográfica (ver figura 3.25) , realizando de la misma forma peticiones en la forma de URLs. Existe software libre, como gvSIG, GRASS, uDIG, Kosmo y otros, que permite el acceso avanzado a la información remota, con el valor añadido de poder cruzarla con información local y así disponer de una gran variedad de herramientas SIG.

3.2.4.3 **WCS (Web Coverage Service):**

WCS provee de una interfaz para servir coberturas, distinto al servicio WMS ya que éste define un mapa como una representación de la información geográfica en forma de archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador, pero el mapa no consiste en los propios datos. Por el contrario WCS sí que proporciona los propios datos, y de esta manera permite su posterior análisis. El servicio WCS permite por tanto el análisis de datos ráster al igual que el servicio WFS permite el análisis de datos vectoriales.

3.2.5 **Historia de los sistemas de información geográfica.**

Hace aproximadaqmente 15.000 años los hombres de Cro-Magnon pintaban en las paredes animales que cazaban, asociando a estos dibujos trazas lineales que, se piensa, coincidían con las rutas migratorias de esas especies en las paredes de las cuevas de Lascaux (Francia). Este es un ejemplo simple en comparación con las tecnologías modernas que viene a demostrar que estos antecedentes tempranos imitan a dos elementos de los SIG modernos: una imagen asociada con un atributo de información.

El primer ejemplo de Sistema de Información Geográfica tal y como los conocemos hoy en día que funcionó, y un considerable avance con respecto a las aplicaciones cartográficas existentes hasta entonces, es el denominado Canadian Geographical Information System (CGIS). Permitía la superposición de capas de información, la realización de mediciones y las digitalizaciones y escaneos de datos. Comenzó su creación en 1964, desde 1967 ha servido para la realización de inventario y planeamiento de la ocupación del suelo en grandes zonas del país. Se financió por el departamento de agricultura de canadi-

ense. IBM aportó hardware necesario. En su creación se plantearon problemas técnicos y conceptuales muchos de los cuales después se han ido resolviendo, en especial los referentes a las bases de datos su estructura y organización y a los métodos de entrada de la información. Otros sistemas contemporáneos independientes a éste como LUNR (Land Use and Natural Resources Information System, Nueva York 1967), MLMIS (Minnesota Land Management Information System, Minnesota 1969), PIOS (Polygon Information Overlay System, 1971), ORMIS (The Oak Ridge Modelling Information System, 1972) y STORET (STOrage and RETrieval of Data for Water Quality Control System) del Servicio Público de Salud de Estados Unidos de la división de Aguas y Control de contaminación.

A continuación se desarrolla un esquema histórico que resume la fiorma en la que se ha ido planteando una de las cuestiones básicas y fundamentales en un Sistema de Información Geográfica, la referente al modelo de datos a utilizar. Se pueden distinguir las siguientes fases en el desarrollo de las actividades del laboratorio:

Primera fase: Creación del programa SYMAP en 1968. Este programa de cartografía asistida por computador, fué elaborado por el laboratorio para grandes ordenadores (mainframes), y sólo permitía obtener borradores de los mapas, que eran trazados mediante una impresora de líneas a baja resolución. Además no disponía apenas de asistencia para la digitalización de la información espacial ni para su posterior manejo rápido y su almacenamiento. Utilizaba una de las formas más simples de conservar digitalmente la información espacial: la lista de coordenadas. Los programas que utilizan el trazador de curvas (plotter) son una continuación de este enfoque, como el CALFORM (1970). En este caso el nivel de detalle que se obtiene es alto aunque todavía lejos de obtener un nivel similar al conseguido mediante el dibujo manual. De todos modos todavía no se había conseguido la elaboración intuitiva del mapa, ni había facilidades para la captación automática ni semiautomática de los datos. Aparecieron en esta época también los programas GRID e IMGRID que se basaban en la representación ráster del espacio geográfico.

Segunda fase: Con el desarrollo de POLYVRT se plantea una im-

portante novedad en cuanto a la forma de estructurar la información espacial. Se integra en ella, explícitamente, la topología de objetos cartográficos. Este nuevo planteamiento tuvo un importante precedente en la estructura de los ficheros que se crearon para la información socioeconómica y demográfica llevada a cabo por la oficina del censo de Estados Unidos. Merece especial mención el formato DIME (Dual Independent Map Encoding), uno de los primeros en incluir explícitamente la topología de la información espacial. Por otro lado, como se ha mencionado ya, el SIG canadiense ya se planteaba y proporcionaba una primera solución al respecto.

Tercera fase: Posteriormente se crea el primer y verdadero sistema de información geográfico de representación vectorial. El programa ODISSEY del Laboratorio de Harvard. Un precedente importante de este sistema de información geográfica fue POLYVRT. Ya incluye la digitalización semiautomática de los datos espaciales, la elaboración interactiva de mapas y la gestión de la base de datos. Actualmente la tendencia consiste en la mejora de las técnicas de reproducción incluyendo el color y formas más expresivas de los mapas. Se usa una estructura de datos topológica que es una versión de la llamada Arco-Nodo que se ha convertido en uno de los modos más aceptados del mercado actual de los SIG. Junto a los desarrollos ya mencionados, en el mismo laboratorio de Harvard se trabajó en la elaboración de una nueva línea de programas cartográficos diferentes que se basan en la representación raster como los citados GRID e IMGRID. De ellos surge el sistema MAP de Dana Tomlin (Yale), que ha sido la base de programas como ERDAS o IDRISI. En contrapartida a las universidades, la empresa comercial ESRI, partiendo de los trabajos de Harvard, desarrolló y amplió sistemas de información geográfica tanto raster, por ejemplo GRID y como vectoriales como PIOS y ARCINFO.

3.2.6 Sistemas de información geográfica en el mercado

3.3 El sistema de información geográfica gvSIG

Como encontramos en la web oficial sel SIG:

gvSIG Desktop es un Sistema de Información Geográfica (SIG), es-

Software SIG	Windows	Mac OS X	GNU/Linux	BSD	Unix	Entorno Web	Llicencia de software
ABACO DbMAP	Sí	Sí	Sí	Sí	Sí	Java	Software no libre
ArcGIS	Sí	No	Sí	No	Sí	Sí	Software no libre
Autodesk Map	Sí	No	No	No	No	Sí	Software no libre
Capaware	Sí (C++)	No	No	No	No	No	Libre: GNU GPL
Caris	Sí	No	No	No	No	Sí	Software no libre
CartaLinx	Sí	No	No	No	No	No	Software no libre
Geomedia	Sí	No	No	No	Sí	Sí	Software no libre
GeoPista	Java	Java	Java	Java	Java	Sí	Libre: GNU
GeoServer	Sí	Sí	Sí	Sí	Sí	Java	Libre: GNU
GRASS	Sí	Sí	Sí	Sí	Sí	Mediante pyWPS	Libre: GNU
gvSIG	Java	Java	Java	Java	Java	No	Libre: GNU
IDRISI	Sí	No	No	No	No	No	Software no libre
ILWIS	Sí	No	No	No	No	No	Libre: GNU
Generic Mapping Tools	Sí	Sí	Sí	Sí	Sí	Sí	Libre: GNU
JUMP	Java	Java	Java	Java	Java	No	Libre: GNU
Kosmo	Java	Java	Java	Java	Java	En desarrollo	Libre: GNU
LocalGIS	Java	Java	Java	Java	Java	Sí	Libre: GNU
LatinoGis	Sí	No	No	No	No	Sí	Software no libre
Manifold	Sí	No	No	No	No	Sí	Software no libre
MapGuide Open Source	Sí	Sí	Sí	Sí	Sí	LAMP/WAMP	Libre: LGNU
MapInfo	Sí	No	Sí	No	Sí	Sí	Software no libre
MapServer	Sí	Sí	Sí	Sí	Sí	LAMP/WAMP	Libre: BSD
Maptitude	Sí	No	No	No	No	Sí	Software no libre
MapWindow GIS	Sí (ActiveX)	No	No	No	No	No	Libre: MPL
Bentley Map	Sí	Abandonado	No	No	Abandonado	Sí	Software no libre
Quantum GIS	Sí	Sí	Sí	Sí	Sí	Sí	Libre: GNU
SAGA GIS	Sí	Sí	Sí	Sí	Sí	No	Libre: GNU
GE Smallworld	Sí	?	Sí	?	Sí	Sí	Software no libre
SavGIS	Sí	No	No	No	No	Integración con Google Maps	Software no libre: Freeware
SEXTANTE	Java	Java	Java	Java	Java	No	Libre: GNU
SITAL	Sí	No	No	No	No	Integración con Google Maps	Software no libre
SPRING	Sí	No	Sí	No	Solaris	No	Software no libre: Freeware
SuperGIS	Sí	No	No	No	No	Sí	Software no libre
TatukGIS	Sí	No	No	No	No	?	Software no libre
TNTMips	Sí	No	No	No	Sí	Sí	Software no libre
TransCAD	Sí	No	No	No	No	Sí	Software no libre
uDIG	Sí	Sí	Sí	No	No	No	Libre: LGNU
GeoStratum	Sí (Flex/Java)	Sí (Flex/Java)	Sí (Flex/Java)	Sí (Flex/Java)	Sí (Flex/Java)	Sí (Flex/Java)	Software no libre

Figura 3.26: Tabla comparativa de diferentes GIS (fuente wikipedia).

to es, una aplicación de escritorio diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas, la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión. Se caracteriza por disponer de una interfaz amigable, siendo capaz de acceder a los formatos más comunes, tanto vectoriales como ráster y cuenta con un amplio número de herramientas para trabajar con información de naturaleza geográfica (herramientas de consulta, creación de mapas, geoprocesamiento, redes, etc.) que lo convierten en una herramienta ideal para usuarios que trabajen con la componente territorial. (fuente: http://www.gvsig.org/)

El proyecto gvSIG surge a través de la Conselleria de Infraestructuras y Transporte por iniciativa de la Generalitat Valenciana (mediante un concurso público). La Universidad Jaume I realiza las tareas de supervisión del proyecto con el objetivo de que el desarrollo siga todos los estándares internacionales (Open GIS Consortium, ver capítulo 3.2.4). IVER Tecnologías de la Información S.A. es la empresa ganadora del concurso que lleva el principal peso del desarrollo.

Actualmente en un contexto de crisis económica surge la Asociación gvSIG. Ésta tiene como objetivo la sostenibilidad del proyecto gvSIG y el desarrollo de la Geomática Libre. Ellos se presentan así: *Entorno a los valores democráticos y solidarios propios del Software Libre plantea el desarrollo de un nuevo modelo de negocio basado en la Cooperación y el Conocimiento compartido donde parte del beneficio generado revierta en el fortalecimiento del Proyecto gvSIG. La sostenibilidad del proyecto gvSIG se fundamenta en el mantenimiento de la Estructura Profesional gvSIG y las infraestructuras necesarias de la Comunidad gvSIG.*

El sistema gvSIG está orientado a usuarios finales de información con naturaleza geográfica, profesionales, estudiantes o personal de admin-

istraciones públicas de cualquier parte del mundo conforman su comunidad de usuarios. Actualmente dispoble en varios idiomas: castellano, valenciano, inglés, alemán, checo, francés, chino, euskera, gallego, italiano, polaco, portugués y rumano. Desarrollado en el lenguaje de programación Java. Funciona en los sistemas operativos Linux, Mac OS X y Windows. Hace uso de librerías estándar de GIS reconocidas, como son Geotools o Java Topology Suite (JTS).

Figura 3.27: Logo GNU GPL.

Distribuido bajo licencia GNU GPL [6] lo que permite su libre uso, distribución, estudio y mejora. Una de sus características más importantes debido a su naturaleza de software libre (open source), es su componente I+D+I. Otra de las características relevantes es la extensibilidad del proyecto. De este modo los posibles desarrolladores pueden ampliar las funcionalidades de la aplicación fácilmente, así como desarrollar aplicaciones totalmente nuevas a partir de las librerías utilizadas en gvSIG (siempre que cumplan la licencia GPL).

Con el proyecto que presenta esta memoria Dielmo 3D S.L. ha ampliado la funcionalidad de gvSIG para dotarlo de la capacidad de lectura y escritura de datos LiDAR y en este proyecto se va a ampliar nuevamente la funcionalidad de este SIG para permitir una gestión estructurada de los proyectos LiDAR y dotarlo de una serie de herramientas de gran utilidad para este tipo de proyectos. De la misma forma que se piensa en la extensibilidad de gvSIG se ha pensado en la de este proyecto. Así la comunidad de usuarios de datos LiDAR podrán crear nuevas herramientas y ampliar la funcionalidad de éste.

El sistema gvSIG permite la integración en una vista tanto datos locales (ficheros, bases de datos) como remotos a través de estándares OGC como son servicios WMS, WFS, WCS o JDBC (Java Database Connectivity). Está diseñado para ser fácilmente extensible, permi-

[6] General Public License o Licencia Pública General de GNU es una licencia creada por la Free Software Foundation en 1989 orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. (Wikipedia)

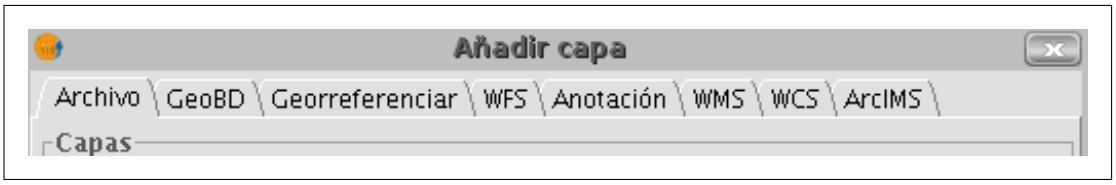


Figura 3.28: Menú añadir capa de gvSIG.

tiendo así una mejora continua de la aplicación y permite el desarrollo de soluciones a medida.

Los proyectos tienen extensión *.gvp*. Este archivo no contiene datos espaciales ni atributos asociados en forma de tablas. Almacena referencias al lugar donde se encuentran dichas fuentes de datos (la ruta a los archivos, los parámetros de conexión a bases de datos, URL de fuentes de datos de internet). Si los datos cambian, los cambios se reflejarán en todos los proyectos que hagan referencia a los mismos.

Las preferencias del sistema pueden ser configuradas. Los parámetros pueden ser especificados en el panel de preferencias. Algunos de estos parámetros configurables son: edición (Color de la selección, Color del eje de referencia, Color de la geometría de la selección, Color del handler de selección), el idioma, el navegador web por defecto (para cualquier búsqueda que se realice desde gvSIG a cualquiera de los hiperenlaces que encontramos dentro de la aplicación), la apariencia, las carpetas donde buscar proyectos (.gvp), datos (ráster o vectoriales) o plantillas (.gvt), la configuración de la pantalla, personalización de los documentos mapas, preferencias de las anotaciones, configuración del firewall, el proxy, el documento vista (proyección, factores de zoom, color, color de selección, unidades de mapa y medida).

3.3.1 **Formatos soportados**

Entre los formatos vectoriales que soporta encontramos: *SHP, DXF, GML, DWG, DGN, KML.*

Entre los formatos ráster que soporta encontramos: *BMP, GIF, TIFF, JPEG, JP2, PNG, VRT, DAT of ENVI, ERDAS (LAN, GIS, IMG), PCI Geomatics (PIX, AUX), ADF of ESRI, ILWIS (MPR, MPL), MAP of PC Raster, ASC, PGM, PPM, RST of IDRISI, RMF, NOS,*

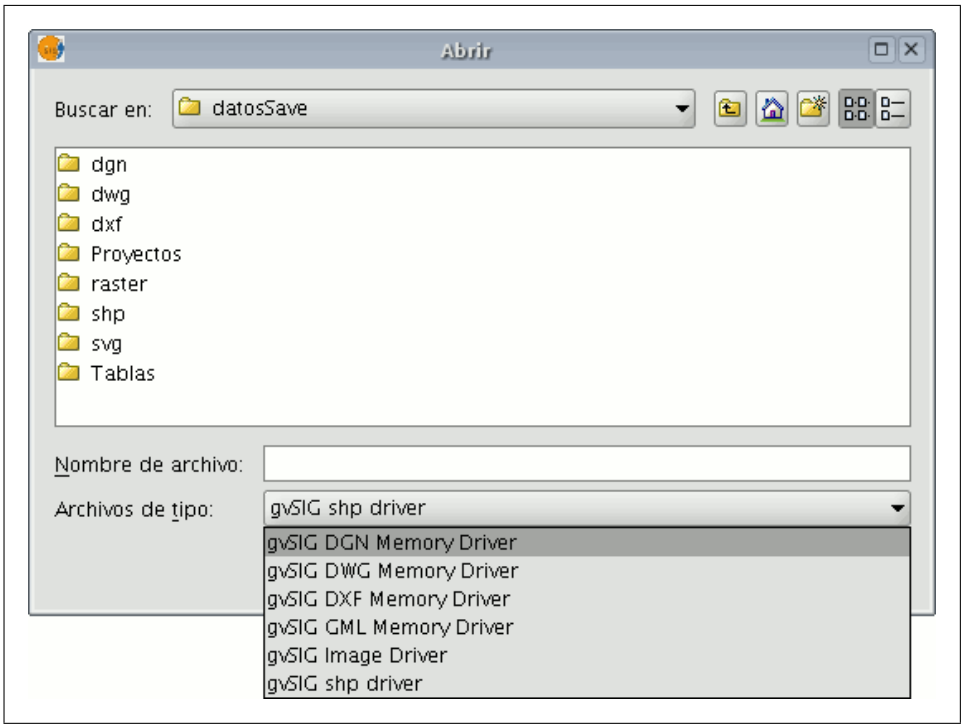


Figura 3.29: Menú añadir capa de gvSIG.

KAP, HDR, RAW.

Entre los servicios remotos a los que puede acceder encontramos: *OGC (WMS, WFS, WCS, WFS-T, WPS), ArcIMS, Ecwp.*

Entre las bases de datos y tablas a las que puede acceder encontramos: *PostGIS, MySQL, ArcSDE, Oracle, JDBC, CSV.*

3.3.2 Herramientas disponibles

Según su naturaleza encontramos diferentes grupos de herramientas en gvSIG entre los que destacan:

- **Navegación**: zooms, desplazamiento, gestión de encuadres, localizador.

- **Consulta**: información, medir distancias, medir áreas, hiperenlace.

Figura 3.30: Herramientas de navegación.

- **Selección**: por punto, por rectángulo, por polígono, por polilínea, por círculo, por área de influencia, por capa, por atributos, invertir selección, borrar selección.

- **Búsqueda**: por atributo, por coordenadas.

- **Geoprocesos**: área de influencia, recortar, disolver, juntar, envolvente convexa, intersección, diferencia, unión, enlace espacial, translación 2D, reproyección, geoprocesos Sextante.

- **Edición gráfica**: añadir capa de eventos, snapping, rejilla, flatness, pila de comandos, deshacer/rehacer, copiar, simetría, rotar, escalar, desplazar, editar vértice, polígono interno, matriz, explotar, unir, partir, autocompletar polígono, insertar punto, multipunto, línea, arco, polilínea, polígono, rectángulo, cuadrado, círculo, elipse.

- **Edición alfanumérica**: modificar estructura tabla, editar registros, calculadora de campos.

- **Servicio de catálogo y nomenclátor.**

- **Representación vectorial:** símbolo único, cantidades (densidad de puntos, intervalos, símbolos graduados, símbolos proporcionales), categorías (expresiones, valores únicos), múltiples atributos, guardar/recuperar leyenda, editor de símbolos, niveles de simbología, bibliotecas de símbolos.

- **Representación ráster**: brillo, contraste, realce, transparencia por píxel, opacidad, tablas de color, gradientes.

- **Etiquetado**: etiquetado estático, etiquetado avanzado, etiquetado individual.

- **Tablas**: estadísticas, filtros, orden ascendente/descendente, enlazar, unir, mover selección, exportar, importar campos, codificación, normalización.

- **Constructor de mapas**: composición de página, inserción de elementos cartográficos (Vista, leyenda, escala, símbolo de norte, cajetín, imagen, texto, gráfico), herramientas de maquetación (alinear, agrupar/desagrupar, ordenar, enmarcar, tamaño y posición), grid, plantillas.

- **Impresión**: impresión, exportación a PDF, a Postcript, a formato de imagen.

- **Redes**: topología de red, gestor de paradas, costes de giro, camino mínimo, conectividad, árbol de recubrimiento mínimo, matriz orígenes-destinos, evento más cercano,área de servicio.

- **Ráster y teledetección**: estadísticas, filtrado, histograma, rango de escalas, realce, salvar a raster, vectorización, regiones de interés, componentes generales, georreferenciación, geolocalización, clasificación supervisada, cálculo de bandas, perfiles de imagen, árboles dedecisión, componentes principales, tasselep cap, fusión de imágenes, diagramas de dispersión, mosaicos.
- **Publicación**: WMS, WFS, WCS de MapServer, WFS de Geoserver.

- **3D y animación**: Vista 3D plana y esférica, capas 3D, simbología 3D, extrusión, edición de objetos 3D, encuadres 3D, animación 2D y 3D, visualización estéreo (anaglifo, horizontal split).

- **Topología**: construcción topológica, edición topológica, generalizar, suavizar, invertir sentido de líneas, convertir capa de líneas/polígonos

a puntos, convertir capa de polígonos a líneas, triangulación de Delaunay/Poligonación de Thiessen, build, clean, correcciones topológicas en modo Batch.

- **Otros**: gestión de Sistemas de Referencia Coordenados, exportar/importar WMC, scripting, gestión de traducciones.

3.3.3 La interfaz de gvSIG

La interfaz gráfica de gvSIG es intuitiva y de fácil manejo. Cualquier usuario familiarizado con los Sistemas de Información Geográfica se familiarizaría rápidamente con el programa. Proporciona los elementos necesarios para comunicarse con el programa.

La interfaz de gvSIG está compuesta por una ventana principal, en la que se encuentran las distintas herramientas, y ventanas secundarias que conforman los documentos propios del programa.

1. Barra de título: Ubicada en la parte superior de la ventana de gvSIG. Contiene el nombre del programa y del proyecto.

2. Botones de maximizar, minimizar la ventana activa del programa, o bien cerrar el programa completamente.

3. Ventana principal: Espacio de trabajo donde se encuentran las distintas ventanas que muestra el *Gestor de proyectos* y los diferentes documentos propios de gvSIG.

4. Barra de menús: Se encuentran agrupadas en forma de menús y submenús las distintas funciones que se pueden realizar con gvSIG.

5. Barra de herramientas: Contiene los iconos de los comandos más habituales y constituyen la forma más fácil y directa de acceder a éstos. Es posible mover su posición inicial en la barra de herramientas haciendo click sobre ellos y arrastrando. Situando el puntero sobre ellos, aparecerá una casilla con la descripción de su función con lo que facilita su aprendizaje.

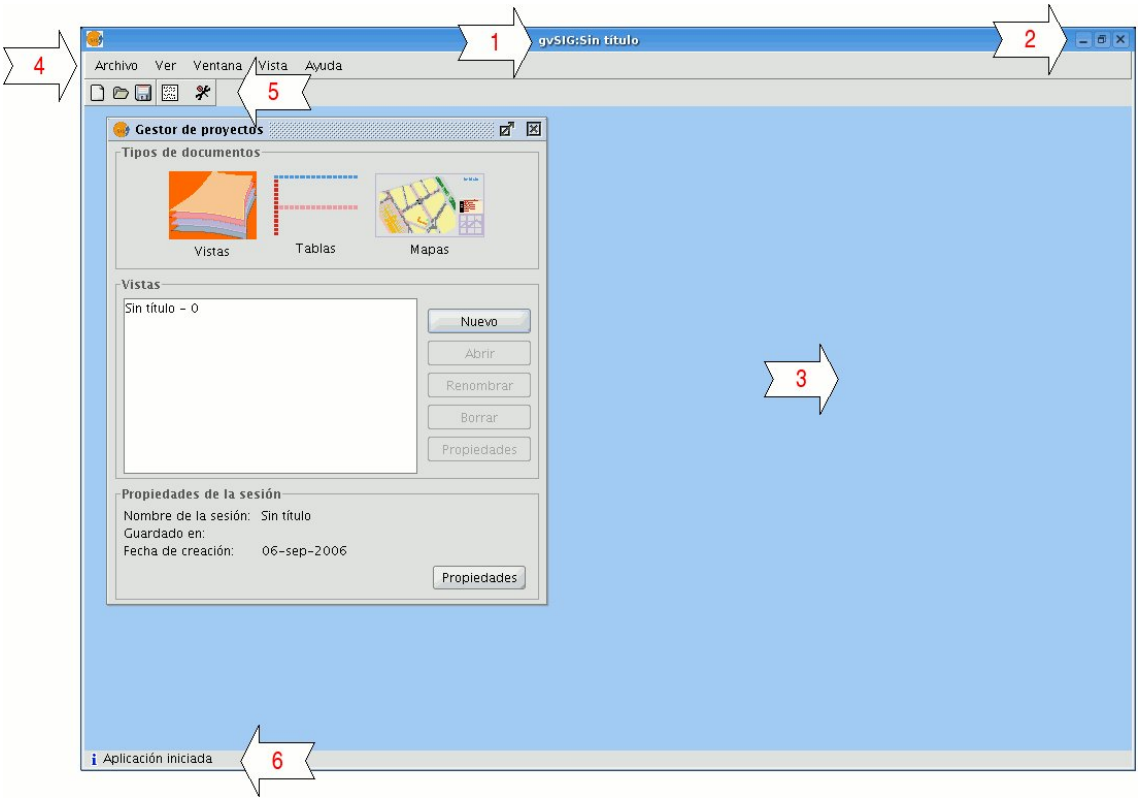


Figura 3.31: Interfaz principal de gvSIG.

6. Barra de estado: Aporta información sobre coordenadas, distancias, etc.

3.3.4 Proyectos y Documentos propios de gvSIG

En el programa gvSIG toda la actividad se localiza en un proyecto. Éste está formado por diferentes documentos. Hay tres tipos de documentos en gvSIG: Vistas, tablas y mapas.

- **Vistas:** Documentos donde se trabaja con los datos gráficos.
- **Tablas**: Documentos donde se trabaja con los datos alfanuméricos.
- **Mapas**: Constructor de mapas que permite insertar los distintos elementos cartográficos que componen un plano como son vista,

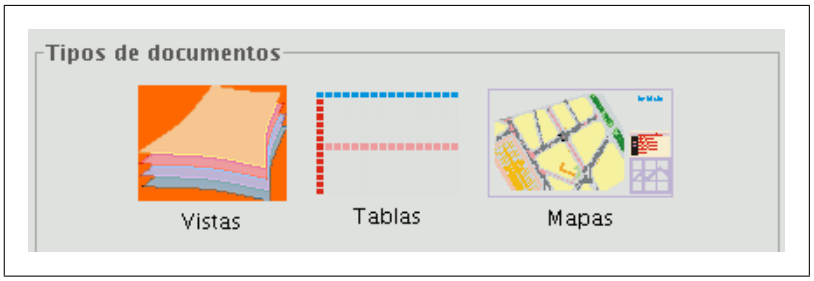


Figura 3.32: Tipos de documentos en gvSIG.

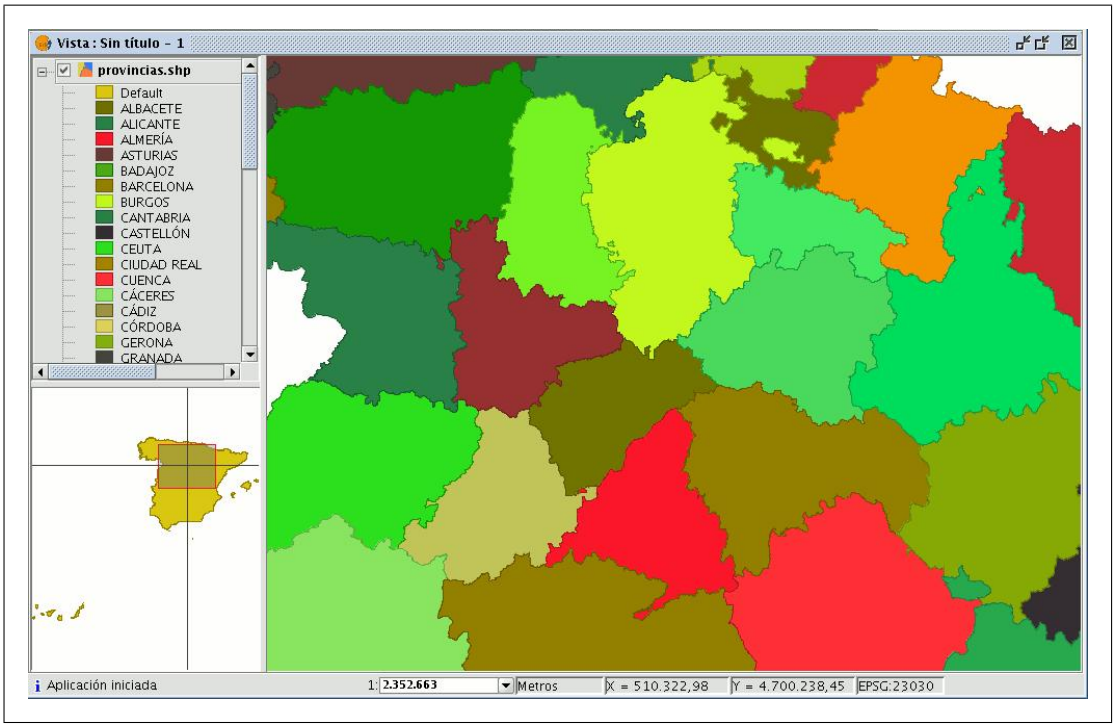


Figura 3.33: Documento vista en gvSIG.

leyenda, escala, norte, etc.

3.3.4.1 **Vista**

Las vistas se tratan de los documentos de gvSIG que constituyen el área de trabajo para el manejo de la información cartográfica.

En un documento de tipo vista pueden existir de modo simultáneo distintas capas de información geográfica de naturaleza muy diversa: hidrografía, comunicaciones, catastro, ortofotos, divisiones administrativas, curvas de nivel, etc. Una vista aparece en gvSIG como una nueva ventana en la que se distinguen los siguientes componentes:

1. Tabla de contenidos (TOC): Situada en la parte izquierda de la ventana. En el TOC se listan todas las capas que contiene la vista y la simbología empleada en la representación de los elementos que componen cada capa.

Figura 3.34: Tabla de contenidos de una vista.

2. Ventana de visualización: Situada en la parte derecha de la ventana de la vista. Es el lugar donde se presentan los datos cartográficos de la vista.

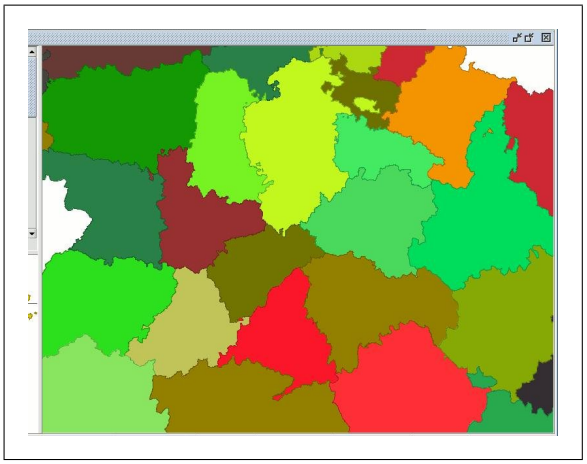


Figura 3.35: Ventana de visualización de una vista.

3. Localizador: Situado en la parte inferior izquierda. Permite situar el encuadre actual en el área de trabajo.

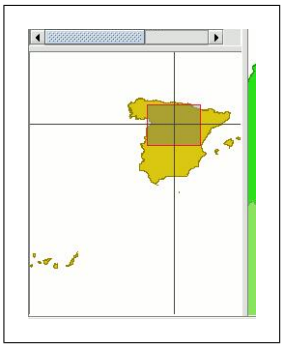


Figura 3.36: Localizador de una vista.

Al abrir una vista, aumenta el número de menús y botones de la ventana principal. Se añaden así las herramientas que permiten trabajar con la vista y los elementos que la conforman. El tamaño del TOC se puede cambiar para permitir una cómoda visualización al completo la descripción de los temas.

A una vista se le puede agregar diferentes tipos de capas de información cartográfica. Puede cargar ficheros vectoriales o ráster.En cada uno de estos grupos puede encontrar una gran variedad de formatos que se han comentado anteriormente (ver capítulo 3.3.1).

El SIG gvSIG es una herramienta que permite realizar cartografía temática con relativa facilidad mediante el *editor de leyendas*. La simbología o representación de los datos o variables de los elementos de una capa permite la selección del color, el tramado, etc. más adecuado para cada uno de ellos.

Es posible la selección de la proyección de una vista. Mediante una ventana se escoge el crs [7]de la vista.

Desde gvSIG puede crear una nueva capa en los formatos shp, dxf y postgis. También mediante la herramienta *Añadir capa de eventos* es posible la creción de una nueva capa en gvSIG partiendo de una tabla.

Es posible la creación de índices espaciales sobre capas vectoriales que aceleren la visualización de la capa cargada en la vista porque

[7]Coordinate Reference System. Sistemas de referencia

la vista utilizará ese índice para cargarse. Esta función generará un archivo *.qix*, con el mismo nombre que la capa a la que se asocia y se ubicaráen el directorio de origen de la capa.

En lo relativo a la *simbología o leyendas* se puede elegir entre las siguientes formas de representación:

- **Símbolo Único:** Es el tipo de leyenda por defecto de gvSIG. Representa todos los elementos de un capa usando el mismo símbolo. Es útil cuando es necesario mostrar la localización de una capa más que cualquiera de sus atributos.

- **Valores Únicos:** Cada registro puede ser representado con un símbolo exclusivo dependiendo del valor que adopte en un determinado campo de la tabla de atributos. Es el método más útil a la hora de desplegar datos categóricos, como puede ser el caso de municipios, tipos de suelo, etc.

- **Intervalos:** Representa los elementos de una capa usando una gama de colores. Los intervalos o colores graduados se usan principalmente para la representación de datos numéricos que tienen una progresión de valores, como la población, la temperatura, inversiones realizadas, etc.

- **Etiquetado:** Textos o etiquetas a la vista de forma automática en función de los valores que adopta cada elemento en un determinado campo de su tabla de atributos.

Las opciones que muestra el menú de simbología varían según el capa sea de puntos, líneas o polígonos. A continuación puede ver las opciones que se muestran para un capa de polígonos, que es el que más herramientas de configuración presenta. En todo momento podrá guardar o cargar (recuperar) una leyenda.

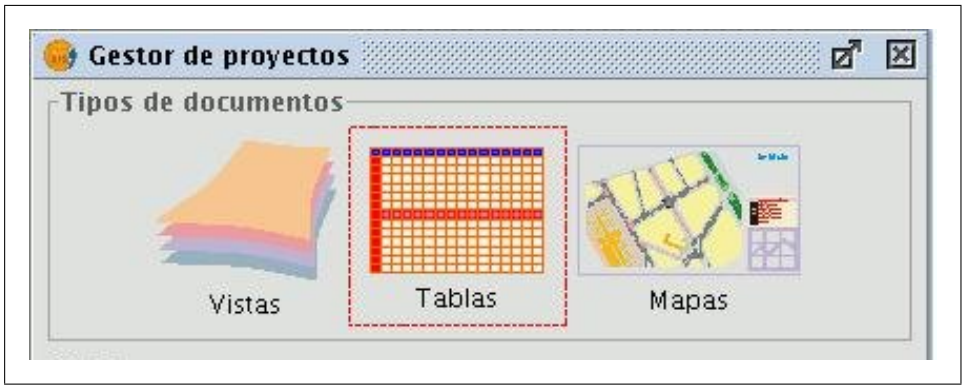


Figura 3.37: Documento tipo tabla en el gestor de proyectos.

3.3.4.2 **Tabla**

Las tablas en gvSIG son documentos que contienen información alfanumérica. Las tablas, como se muestra en la imagen 3.38, están compuestas por:

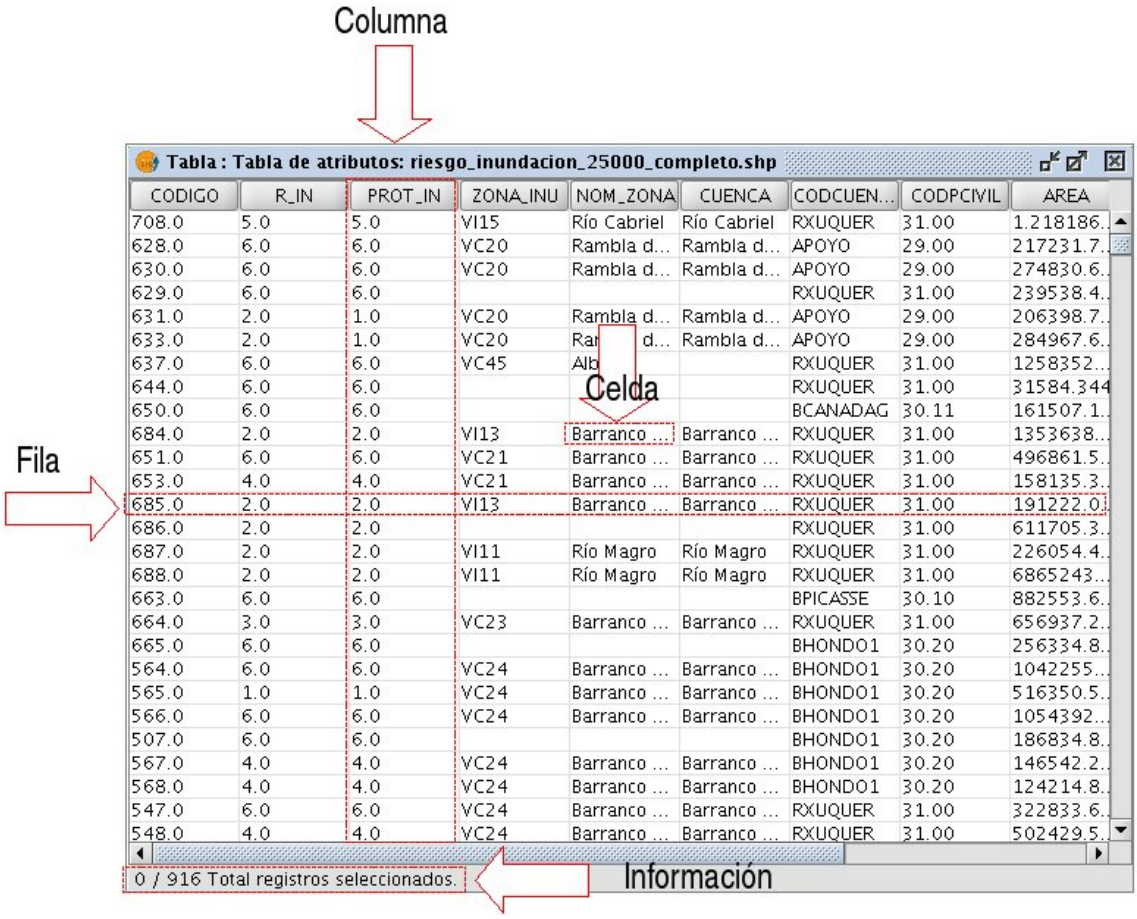
- **1. Fila o registro**: Es la representación de los distintos elementos de la tabla.

- **2. Columna o campo**: Son los tipos de atributos que definen a cada elemento.

- **3. Celda**: La intersección de un registro y un campo es una celda. La celda es el elemento mínimo de trabajo y puede contener información.

- **4. Información de registros**: Informa del total de elementos (registros) que contiene la tabla.

Todas las capas vectoriales tienen su tabla de atributos asociada. Cada elemento gráfico de una capa tiene asociado su correspondiente registro en dicha tabla de atributos. Es posible realizar selecciones tanto simples como multiples de los elementos de la tabla.



CODIGO	R_IN	PROT_IN	ZONA_INU	NOM_ZONA	CUENCA	CODCUEN...	CODPCIVIL	AREA
708.0	5.0	5.0	VI15	Río Cabriel	Río Cabriel	RXUQUER	31.00	1.218186..
628.0	6.0	6.0	VC20	Rambla d...	Rambla d...	APOYO	29.00	217231.7..
630.0	6.0	6.0	VC20	Rambla d...	Rambla d...	APOYO	29.00	274830.6..
629.0	6.0	6.0				RXUQUER	31.00	239538.4..
631.0	2.0	1.0	VC20	Rambla d...	Rambla d...	APOYO	29.00	206398.7..
633.0	2.0	1.0	VC20	Ra d...	Rambla d...	APOYO	29.00	284967.6..
637.0	6.0	6.0	VC45	Alb		RXUQUER	31.00	1258352...
644.0	6.0	6.0				RXUQUER	31.00	31584.344
650.0	6.0	6.0				BCANADAG	30.11	161507.1..
684.0	2.0	2.0	VI13	Barranco ...	Barranco ...	RXUQUER	31.00	1353638...
651.0	6.0	6.0	VC21	Barranco ...	Barranco ...	RXUQUER	31.00	496861.5..
653.0	4.0	4.0	VC21	Barranco ...	Barranco ...	RXUQUER	31.00	158135.3..
685.0	2.0	2.0	VI13	Barranco ...	Barranco ...	RXUQUER	31.00	191222.0..
686.0	2.0	2.0				RXUQUER	31.00	611705.3..
687.0	2.0	2.0	VI11	Río Magro	Río Magro	RXUQUER	31.00	226054.4..
688.0	2.0	2.0	VI11	Río Magro	Río Magro	RXUQUER	31.00	6865243...
663.0	6.0	6.0				BPICASSE	30.10	882553.6..
664.0	3.0	3.0	VC23	Barranco ...	Barranco ...	RXUQUER	31.00	656937.2..
665.0	6.0	6.0				BHONDO1	30.20	256334.8..
564.0	6.0	6.0	VC24	Barranco ...	Barranco ...	BHONDO1	30.20	1042255...
565.0	1.0	1.0	VC24	Barranco ...	Barranco ...	BHONDO1	30.20	516350.5..
566.0	6.0	6.0	VC24	Barranco ...	Barranco ...	BHONDO1	30.20	1054392...
507.0	6.0	6.0				BHONDO1	30.20	186834.8..
567.0	4.0	4.0	VC24	Barranco ...	Barranco ...	BHONDO1	30.20	146542.2..
568.0	4.0	4.0	VC24	Barranco ...	Barranco ...	BHONDO1	30.20	124214.8..
547.0	6.0	6.0	VC24	Barranco ...	Barranco ...	RXUQUER	31.00	322833.6..
548.0	4.0	4.0	VC24	Barranco ...	Barranco ...	RXUQUER	31.00	502429.5..

Figura 3.38: Tabla en gvSIG.

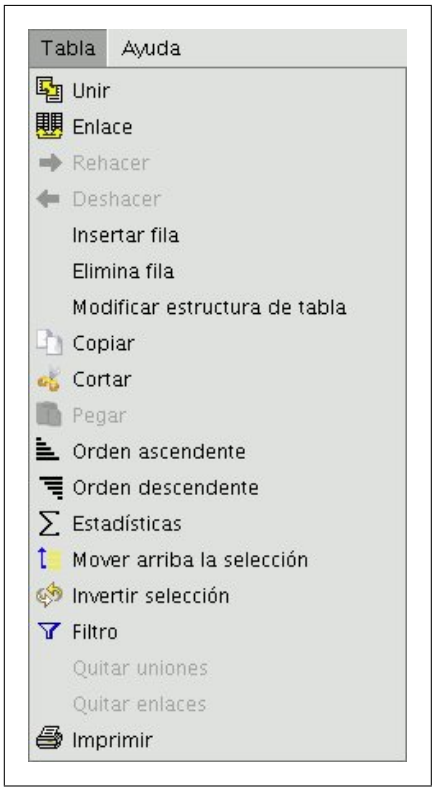


Figura 3.39: Herramientas del menu Tabla.

3.3.4.3 Mapa

En gvSIG los documentos de tipo mapa permiten diseñar y combinar en una página los elementos que se deseen para que aparezcan en un mapa impreso. El acceso documento mapa se hace a través del gestor de proyectos de gvSIG de la misma forma que en los demás tipos de documentos. Los mapas tienen ciertas propiedades configurables:

- Malla: la malla permite que cualquier elemento insertado en el mapa se ajustará a la misma. Al establecerla hay que tener en cuenta las propiedades:
  - El espaciado horizontal y vertical de la malla que definen la separación entre puntos que componen la malla.
  - El tamaño de salida del documento escogido (A2, A3, A4, ...).
- Regla: activándola se visualizará una regla que puede proporciona ayuda al dibujo.
- Editable: Esta opción permite bloquear los objetos que conforman el mapa, impidiendo que se puedan realizar modificaciones.

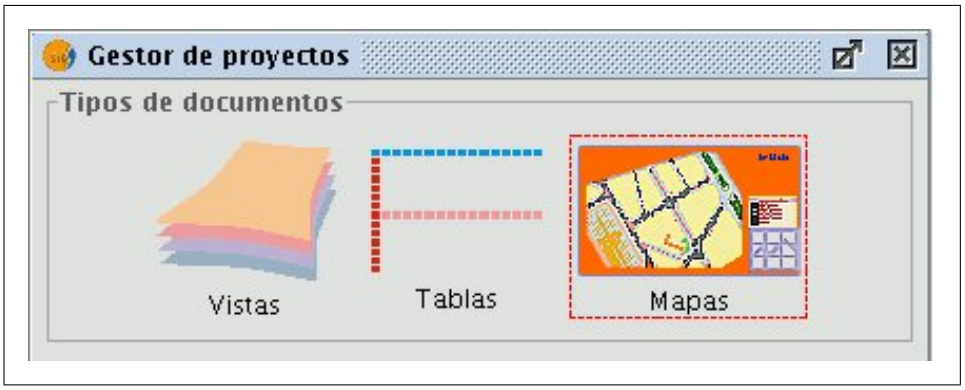


Figura 3.40: Documento tipo mapa en el gestor de proyectos.

- Página: es posible definir el espacio de trabajo, es decir, el tamaño y propiedades de la página donde se va a realizar la composición del mapa. Destacan los siguientes parámetros:
  - Tamaño de página: El desplegable le permite definir el origen y el tamaño del papel donde va a ser impreso el mapa. Puede seleccionar un tamaño estándar o definir uno propio.
  - Unidades de medida: Puede seleccionar las unidades de medida de la Altura y Anchura de página.
  - Orientación: Establece la orientación del papel, horizontal o vertical.
  - Márgenes: Permite definir los cuatro márgenes de la hoja. La regla se ajusta a los márgenes de la página.
  - Resolución del resultado: Puede escoger entre resolución alta, baja y normal.

En gvSIG es posible incorporar a un mapa los siguientes elementos cartográficos:

- Vistas: algunos elementos cartográficos están íntimamente ligados al documento vista de manera que al realizar cambios en ésta (cambios de zoom, desplazamientos, modificación de leyendas, organización de capas, etc.) se ven reflejados en el mapa. Este vínculo es configurable.
- Imágenes: Esta herramienta le permite insertar una imagen en el mapa.

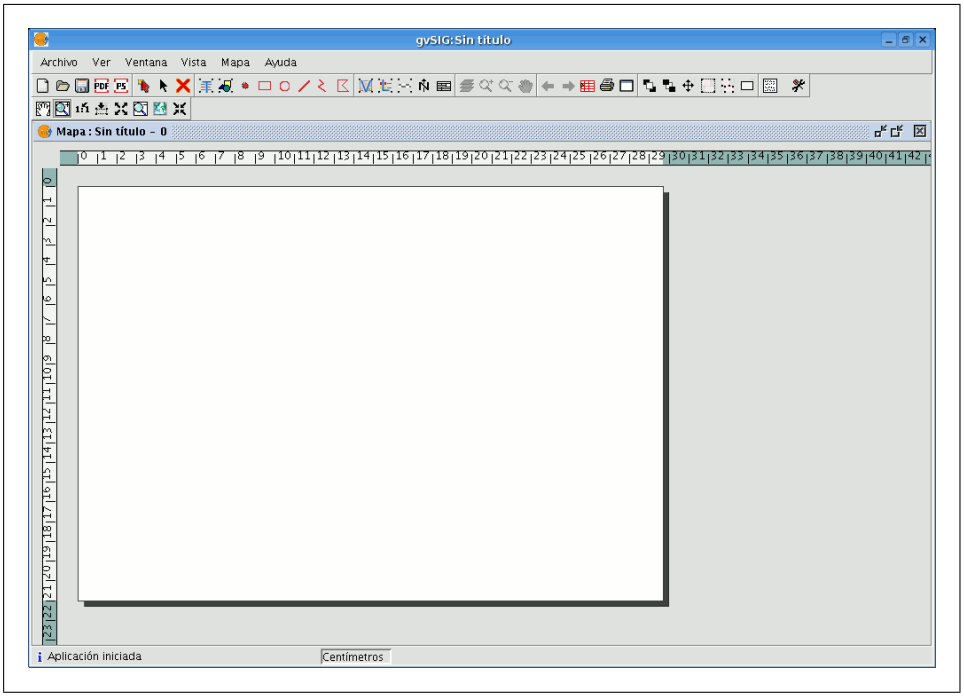


Figura 3.41: Documento tipo mapa.

- Barras de escala: esta herramienta le permite insertar una escala (relacionada con una vista) en el mapa.

- Leyendas: La leyenda representa las capas visibles de la tabla de contenidos (TOC) de la vista seleccionada. Si se inserta una leyenda en el mapa ésta se añade en el mismo orden en el que aparece en el TOC.

- Objetos gráficos: Puede insertar los siguientes tipos de elementos gráficos:
  - Puntos
  - Rectángulos
  - Círculos
  - Líneas
  - Polilíneas
  - Polígonos

- Norte: puede insertar un símbolo de norte en el mapa.

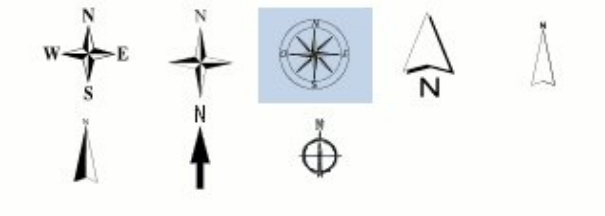


Figura 3.42: Símbolos de norte.

Figura 3.43: Opciones asociadas a un mapa.

- Textos: permite insertar texto en el mapa. Varios parámetros son configurables como la alineación, tipo de fuente, grados de inclinación, marco y el título asociado al marco.

- Cajetines: gvSIG ha incorporado en su barra de herramientas una que le permite insertar un cajetín en el mapa.

El sistema gvSIG permite guardar la configuración de un mapa como plantilla y así poder aprovecharlo en otro momento con diferentes orígenes de datos. Partiendo de un Mapa, es posible guardar la distribución y propiedades de sus elementos.

Es posible exportar la composición realizada a un fichero postScript y/o pdf.

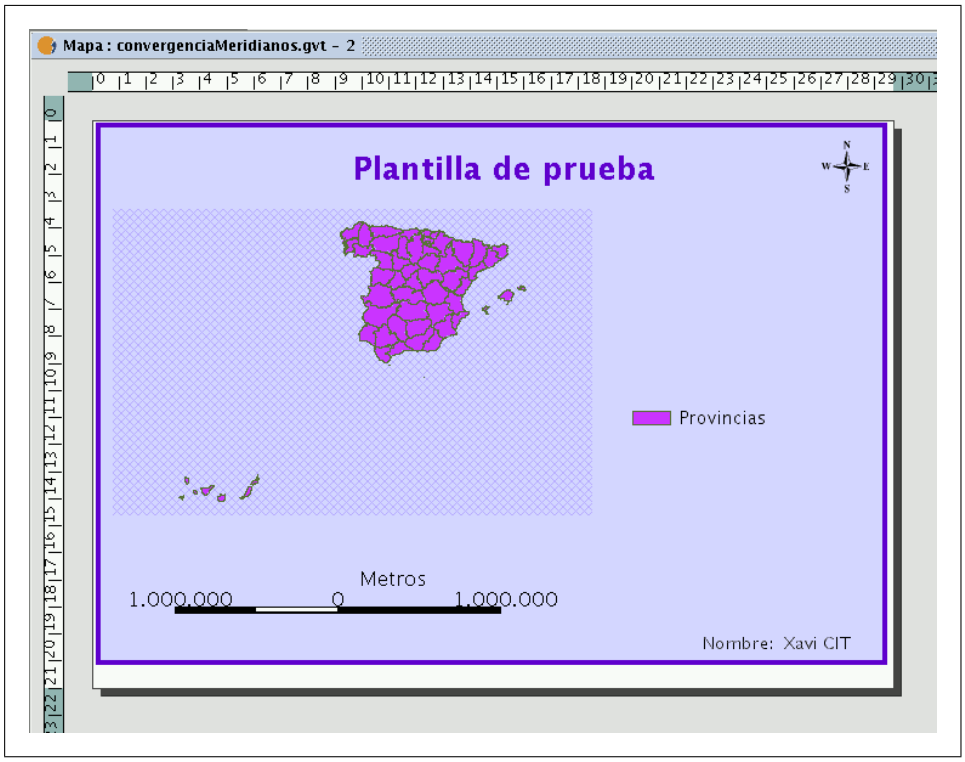


Figura 3.44: Plantilla de un mapa.

3.3.5 Extensión geoBD (gestor de bases de datos)

Una base de datos o banco de datos (en ocasiones abreviada BB.DD. o DB) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. El contenido de una base de datos engloba la información que concierne a una organización. De esta manera la información está disponible para los usuarios. Una finalidad de la base de datos es conseguir que la información no sea redundante. Es por ello que cuando se modela la estructura de una base de datos se aplica un proceso de normalización que pretende eliminar redundancias y estructurar la información de manera óptima. Esta tarea la debe de ralizar personal cualificado.

Una tabla de datos es la unidad lógica de almacenamiento de la información en una base de datos. Una tabla está formada por registros (llamados también filas o tuplas).

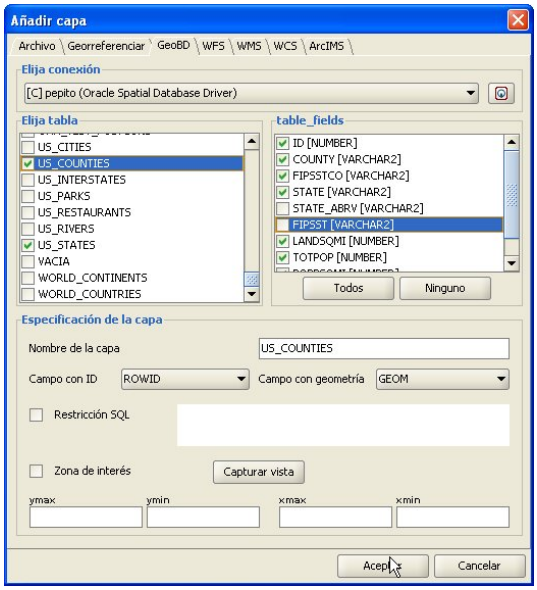
Esta extensión de gvSIG dota al sistema de la posibilidad de acceder a bases de datos geográficas de forma sencilla y unificada para

distintos proveedores. En la actualidad gvSIG soporta los siguientes sistemas gestores de bases de datos:

- PostGIS
- MySQL
- HSQLDB
- Oracle Spatial (SDO Geometry)

(a) Parámetros de conexión a una base de datos.

(b) Solicitud de contraseña.



(c) Ventana añadir geoBD.

Figura 3.45: Interfaces de la extensión geoBD.

3.3.5.1 La extensión espacial de PostgreSQL. PostGIS

PostgreSQL es un potente sistema de base de datos objeto-relacional de código abierto. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad, integridad de datos y corrección. Funciona en todos los principales sistemas operativos, incluyendo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows. Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). Se incluye la mayor parte de SQL: 2008 tipos de datos, incluyendo INTEGER, NUMERIC, Boolean, CHAR, VARCHAR, DATE, INTERVAL, y TIMESTAMP. También soporta almacenamiento de objetos binarios grandes, como imágenes, sonidos o vídeo. Se dispone de interfaces de programación nativo de C / C + +, Java,. Net, Perl, Python, Ruby, Tcl, ODBC, entre otros, y una excelente documentación.

PostGIS ha sido desarrollado para dar soporte a objetos geográficos a bases de datos relacionales PostgreSQL. PostGIS habilita espacialmente al servidor PostgreSQL, permitiendo que sea utilizado como una base de datos espacial para los sistemas de información geográfica (SIG), al igual que la ArcSDE[8] de ESRI o la extensión espacial de Oracle. PostGIS sigue el OpenGIS *Simple Features Specification for SQL* y ha sido certificado como compatible con los tipos y funciones perfil (ver imagen 3.46) .

PostGIS ha sido desarrollado por Refractions Research[9] como un proyecto de tecnología de base de datos espaciales de código abierto. PostGIS es liberado bajo la licencia GNU General Public License. Se sigue desarrollando PostGIS, y se han añadido herramientas de interfaz de usuario, soporte de la topología de base, validación de datos, transformación de coordenadas, APIs de programación y mucho más. La lista de futuros proyectos incluye soporte completo de topología, ráster, redes y enrutamiento, superficies tridimensionales,

[8] Advanced Spatial Data Server. Gestiona datos espaciales en un sistema de gestión de bases relacionales (RDBMS) y le permite tener acceso a clientes de ArcGIS.
[9] (http://www.refractions.net/)

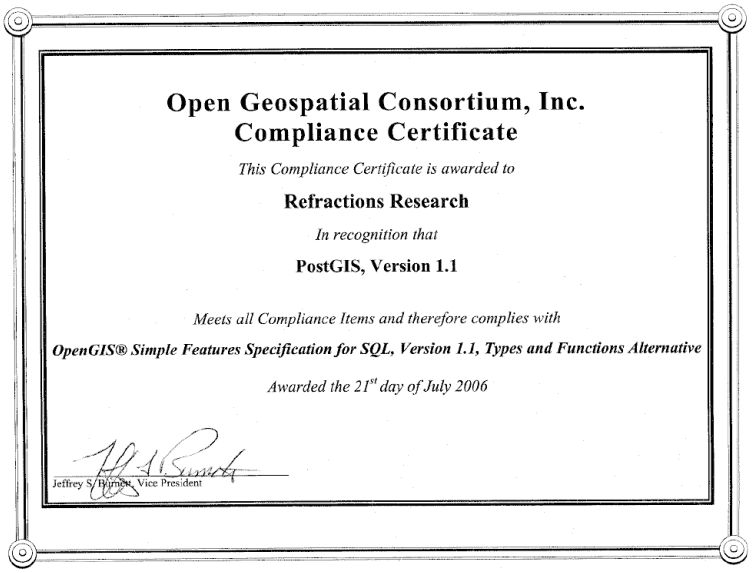


Figura 3.46: Certificación OGC.

curvas y splines y otras características.

PostGIS implementa funciones geométricas y topológicas para poder realizar el tratamiento de datos espaciales basándose en el estándar OGC con el valor añadido de ser de libre distribución.

Entre las funciones más importantes destacan:

- AsText: muestra una geometría representación textual.

- GeometryFromText: convierte un objeto de la representación textual a un objeto geometría.

- AddGeometryColumn: Añade una columna geométrica a una tabla existente.

- Contains: determina si una geometría dada contiene en su interior otra geometría determinada.

- Difference: calcula la diferencia geométrica entre dos geometrías.

- Intersects: determina si dos geometrías se intersectan.
- Perimeter: devuelve el perímetro de una geometría polígono.
- Area: devuelve el área de una geometría dada de tipo polígono.
- Distance: devuelve la distancia mínima entre dos geometrías.
- Touches: determina si dos geometrás dadas son adyacentes.
- Intersection: determina la intersección entre dos geometrías dadas.

Los sistemas de gestión de bases de datos (en inglés database management system, DBMS) son un tipo de software dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. El principal objetivo es proporcionar un entorno para extraer, consultar y manipular la información de manera eficiente de una base de datos. Sus funciones principales son:

- Creación y organización de la base de datos.
- Manejo de datos según las peticiones de los usuarios.
- Acceso a datos rápido.
- Registrar la utilización de la base de datos.
- Interacción con el manejador de archivos.
- Recuperación en caso de fallos.

- Control de concurrencia.

- Seguridad e integridad

Una base de datos pasa a ser una base de datos espacial cuando los datos que contiene estan espacialmente referenciados. Por lo tanto las bases de datos espaciales combinan información de dos tipos: espacial y temática.

3.3.6 **Extensión de Geoprocesamiento**

La extensión de goprocesamiento ha sido desarrollada para la realización de análisis geográfico. gvSIG incluye una herramienta de geoprocesamiento. Ésta permite aplicar una serie de procesos estándar sobre las capas vectoriales cargadas en una vista de gvSIG dando como resultado nuevas capas de información vectorial que aportarán una nueva información adicional a las capas de partida. En la primera versión de la extensión de geoprocesamiento se han implementado los siguientes geoprocesos:

- **Área de influencia (buffer):** Este geoproceso crea una nueva capa vectorial de polígonos, generados como zonas de influencia alrededor de las geometrías de los elementos vectoriales de una capa de entrada. Las geometrías de la capa de entrada pueden ser tanto de puntos, como de líneas o polígonos. Se pueden generar varios anillos concéntricos equidistantes en torno a las geometrías de entrada. Además, en el caso de geometrías de entrada poligonales el área de influencia puede ser exterior, interior o exterior e interior al polígono original. Este Geoproceso puede ser de gran utilidad para la realización de análisis de corredor. Por ejemplo: Qué zonas urbanos no tienen una parada de autobuses en un radio de 500 m, qué zonas urbanas carecen de colegios en un radio de 1000 m, qué pozos incumplen la normativa al no respetar la distancia mínima entre dos consecutivos, etc.

- **Enlace espacial (Spatial Join):** Este geoproceso, al igual que un join entre tablas, permite transferir los atributos de una capa

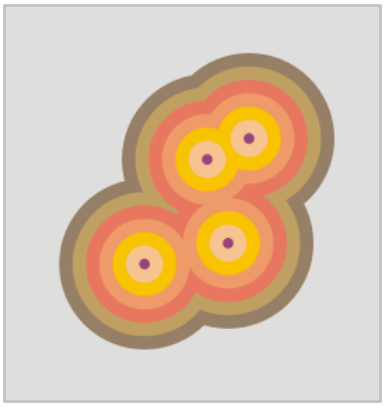


Figura 3.47: Área de influencia de 4 puntos.

a otra en base a una característica común. A diferencia del join de las bases de datos relacionales, en este caso la característica común no es que un campo de las dos tablas tome el mismo valor (la clave del join), sino que los elementos relacionados de las dos capas cumplan unos criterios espaciales. El geoproceso Enlace Espacial implementado por la extensión de geoprocesamiento de gvSIG permite seguir dos tipos de criterios espaciales para establecer el enlace espacial:

- **Vecino más próximo (relación 1-1)**. Asigna a un elemento de la capa origen los atributos del elemento más próximo de la capa enlazada. En el caso de que el elemento más próximo intersecte (o esté contenido para el caso de polígonos) al elemento original, habiendo por tanto varias intersecciones, el algoritmo tomará el primer elemento analizado de las posibles intersecciones.

- **Contenido en (relación 1-M)**. Relaciona un elemento de la capa origen con varios elementos de la capa destino (en concreto, con aquellos que son intersectados). En este caso la capa origen no heredará los atributos de la capa relacionada, sino que la operativa será muy parecida a la del geoproceso Disolver. Para los M elementos relacionados con un elemento de la capa origen, se dará al usuario la posibilidad de escoger una o varias funciones resumen (media, mínimo, máximo, sumatorio) que se aplicarán sobre los atributos numéricos de la capa enlazada.

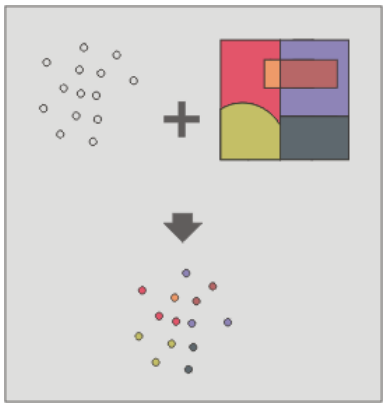
- 

Figura 3.48: Enlace espacial.

- **Recortar (clip):** Este geoproceso permite limitar el ámbito de trabajo de una capa vectorial (da igual que sea de puntos, líneas o polígonos), extrayendo de ésta una zona de interés. Para ello, el usuario deberá proporcionar una capa de entrada (la capa de la que se quiere extraer un zona) y una capa de recorte, de forma que la unión de las geometrías incluidas en la capa de recorte definirán el ámbito de trabajo. El geoproceso recorrerá todos los elementos vectoriales de la capa de entrada (features), y para aquellos que estén contenidos en el ámbito de trabajo definido por la capa de recorte, calculará sus intersecciones, de forma que en la capa resultado solo estarán los elementos vectoriales de nuestro ámbito de interés. La porción de geometría que quede fuera del ámbito de trabajo será recortada. El esquema alfanumérico de la capa de entrada se mantiene intacto.
  Este Geoproceso puede ser de gran utilidad cuando queramos limitar nuestro ámbito de trabajo a una zona concreta de detalle, pero toda la cartografía disponible es de carácter general. Así, por ejemplo, a la hora de montar un GIS Municipal, permitiría incluir cartografía de carácter nacional o regional limitándola al ámbito de interés del municipio.

- **Diferencia:** El geoproceso diferencia trabaja con dos capas: la capa de entrada y la capa de solape. Es conocido como operador NOT espacial. Permite obtener aquellas zonas de una capa que no

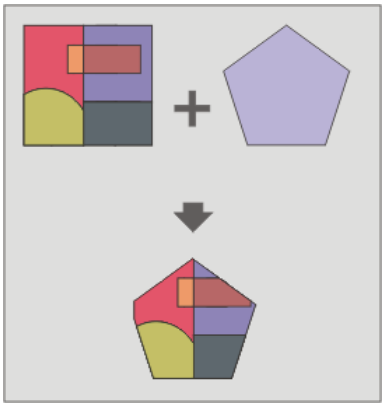


Figura 3.49: Enlace espacial.

están presentes en la otra capa. Las geometrías tanto de la capa de entrada como de la capa de solape deberán ser de polígonos. La capa resultante conservará intacto el esquema alfanumérico de la capa de entrada, pues al fin y al cabo nos viene a dar más información sobre ésta: aquellas zonas que son geométricamente disjuntas de la geometría de la capa de solape.
Este Geoproceso puede ser de gran utilidad en numerosas situaciones. Por ejemplo, se puede considerar como el complemento (contrario) del Geoproceso Recortar/Clip. Si Recortar permite excluir todo aquello que no pertenezca a un ámbito geográfico de estudio, Diferencia permite realizar justamente lo contrario: excluir de nuestra capa de trabajo un determinado ámbito. Esto es de utilidad por ejemplo en el caso de traspaso de competencias territoriales entre diferentes Administraciones. Así, si una Administración Estatal traspasa determinadas competencias a una Regional, puede decidir excluir de sus bases de datos la zona geográfica objeto del traspaso.

- **Intersección:** Este geoproceso opera sobre dos capas, la capa de entrada y la capa de solape, cuyas geometrías han de ser forzosamente poligonales. Para cada geometría de la capa de entrada, calcula la interseccion con las diferentes geometrías de la capa de solape, originando un nuevo elemento por cada intersección. Este elemento tomará todos los atributos alfanuméricos de las geometrías que lo originaron (de entrada y solape). Por este motivo (modela zonas del espacio que cumplen la condición de pertenecer

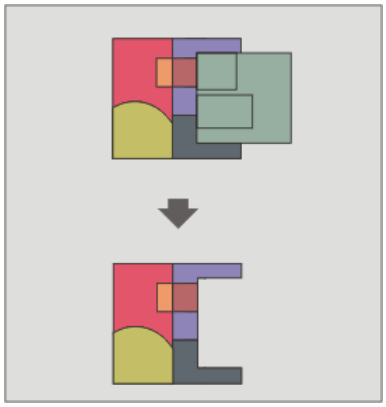


Figura 3.50: Diferencia.

a los dos polígonos que lo han originado) a este geoproceso se le conoce como operador AND espacial.
Un ejemplo de aplicación de este geoproceso sería para, dada una capa de usos del suelo Corine 2000, y una capa del mapa geológico nacional, obtener una capa de polígonos con información homogénea de uso del suelo y material geológico.

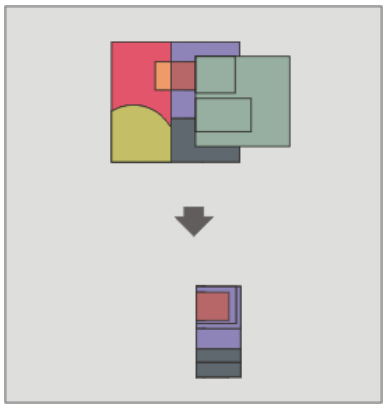


Figura 3.51: Intersección.

- **Unión:** Este geoproceso, al igual que los geoprocesos Intersección y Diferencia opera sobre dos capas de polígonos, obteniendo sus intersecciones (por este motivo, a estos tres geoprocesos se les conoce como geoprocesos de solape). Al geoproceso Unión se le conoce como OR espacial, porque la capa de resultado estará formada por las geometrías que aparecen en las dos capas (intersecciones entre los polígonos), mas las geometrías que aparecen solamente en una u otra de las dos capas puestas en relación. Esto se traduce en que el geoproceso realiza tres pasadas: la primera para

calcular la intersección de ambas capas, la segunda para calcular las diferencias de la primera con la segunda, y la tercera pasada para calcular las diferencias de la segunda capa con la primera. Este Geoproceso siempre que nos interese generar nuevas capas que pongan de manifiesto la ocurrencia de dos fenómenos, de forma que se resalte la ocurrencia de alguno de los dos (o de los dos).

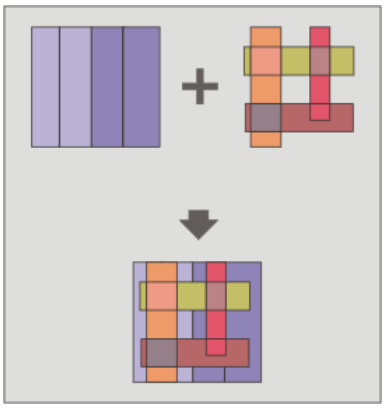


Figura 3.52: Unión.

- **Convex Hull (mínimo polígono convexo):** Este geoproceso cálcula la envolvente convexa (convex hull), o polígono convexo de menor área que envuelve a todos los elementos vectoriales de una capa de entrada. Opera únicamente con una capa de entrada, cuyo tipo de geometría podrá ser de cualquier tipo. Las aplicaciones de este geoproceso pueden ser de distinto tipo: Determinar la zona de cobertura de un determinado fenómeno geográfico. Cálculo del diametro de la zona cubierta por una serie de geometrías, etc.

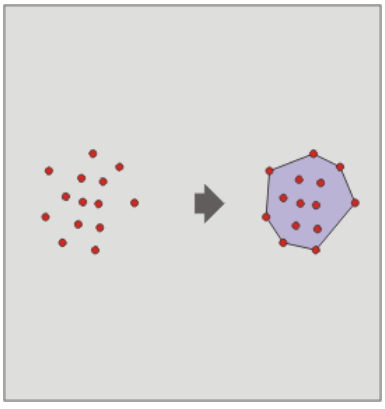


Figura 3.53: Convex Hull.

- **Dissolve (agrupar por adyacencia y criterios alfanuméricos):** Este geoproceso actúa sobre una sola capa de entrada, cuyo tipo de geometría ha de ser forzosamente de polígonos. El proceso analiza cada polígono de la capa de entrada, de tal forma que fusionará en un solo polígono aquellos polígonos que tomen identico valor para un campo especificado. Además, permite introducir el criterio espacial en la decisión de fusionar varios polígonos. De esta forma, podemos seleccionar que para que dos polígonos sean fusionados, además de tomar idéntico valor en el atributo especificado deban ser adyacentes espacialmente. Esto puede ser de utilidad en múltiples situaciones. Supongamos, por poner un ejemplo, que disponemos de una capa de polígonos que representa los municipios de una determinada comunidad autónoma. En este momento necesitamos , para realizar un informe, disponer de una capa de polígonos con las provincias, pero en ese momento no tenemos esta informacion. Podemos generar una capa de provincias lanzando el geoproceso Disolver, especificando que se fusionarán aquellos polígonos que tomen igual valor para el campo PROVCOD -código de provincia.

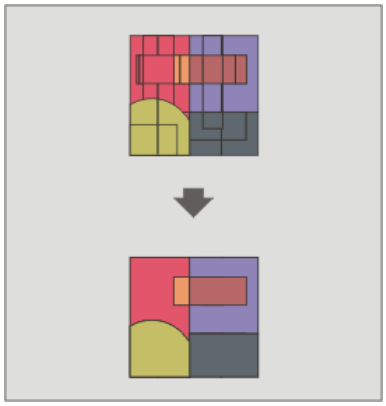


Figura 3.54: Dissolve.

- **Juntar (merge):** Este geoproceso actúa sobre una o varias capas, generando una nueva capa que aúne todas las geometrías de la capa de entrada. La capa resultante de este geoproceso conservará los atributos de una de las capas de entrada, la especificada por el usuario. Del resto de capas no seleccionadas, se conservarán aquellos atributos cuyo nombre y tipo de dato coincida con alguno de los de la capa seleccionada por el usuario. Este geoproceso es

de utilidad, por ejemplo, cuando nos llega una serie cartográfica, separada por hojas, y deseamos juntar el contenido de las diferentes hojas en una sola capa. Tal es el caso de la serie de hojas Magna, publicada por el ITGME español (Instituo Tecnológico y Geominero de España).

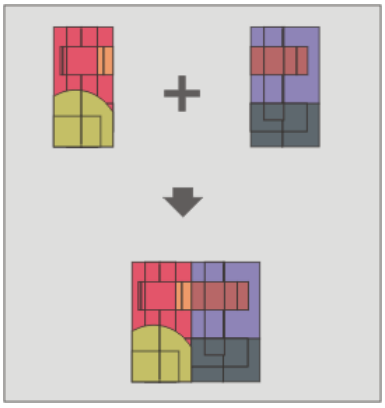


Figura 3.55: Juntar.

- **Traslación 2D:** Este geoproceso permite aplicar una transformación de traslación sobre todos los puntos de las geometrías de la capa de entrada. Para tal fin, el usuario deberá especificar el desplazamiento en X y en Y a aplicar. Este geoproceso puede ser de gran utilidad para hacer concordar cartografías procedentes de fuentes distintas, en lo que se viene a denominar por el término inglés conflation.

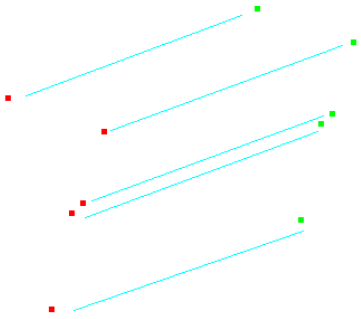


Figura 3.56: Juntar.

- **Reproyección:** Este geoproceso permite cambiar la proyección geodésica de los elementos vectoriales de la capa de entrada. Para tal fín, el usuario deberá especificar la nueva proyección a aplicar. Este geoproceso puede ser de gran utilidad para hacer concordar

cartografías en un mismo proyecto cartografías que se encuentran en proyecciones distintas.

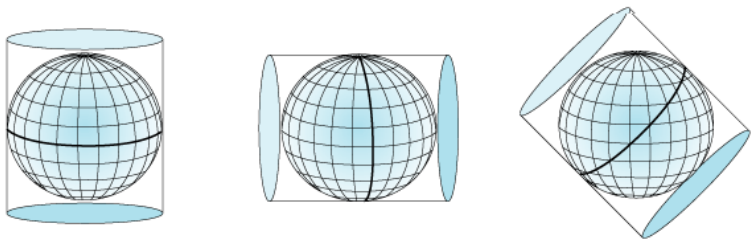


Figura 3.57: Reproyeccion.

3.3.7 **Libreria Java Topology Suite**

JTS o Java Topology Suite es una API Java que implementa un conjunto básico de operaciones de datos espaciales utilizando un modelo de precisión explícita y robustos algoritmos geométricos. JTS está destinada a ser utilizada en el desarrollo de aplicaciones que soportan la validación, la integración y la consulta de bases de datos espaciales. JTS pretende aplicar el OpenGIS Simple Features Specification (SFS) con la mayor precisión posible. En algunos casos trata de elegir una alternativa razonable y coherente al SFS. Estas diferencias de la SFS se aportan en documentación de vivid solutions (http://www.vividsolutions.com).

El diseño del JTS tiene por objeto cumplir los siguientes objetivos:

- El modelo espacial y definicion de métodos se ajustarán a las características OpenGIS.
- El diseño de la API pretende seguir las convenciones de Java en la medida de lo posible.
- Las funciones de JTS apoyarán un modelo de precisión definida. Los algoritmos JTS serán contundentes en virtud del modelo de precisión.

- Los métodos devolverán topológicamente y geométricamente los resultados correctos en los modelos de precisión definidos siempre que sea posible.

- La corrección es la máxima prioridad, el espacio y la eficiencia temporal son importantes, pero secundarios.

- JTS será lo suficientemente rápida para ser utilizada en un entorno de producción.

- Los algoritmos y el código utilizado en JTS serán claros y estarán bien estructurados, para facilitar su comprensión por parte de otros desarrolladores.

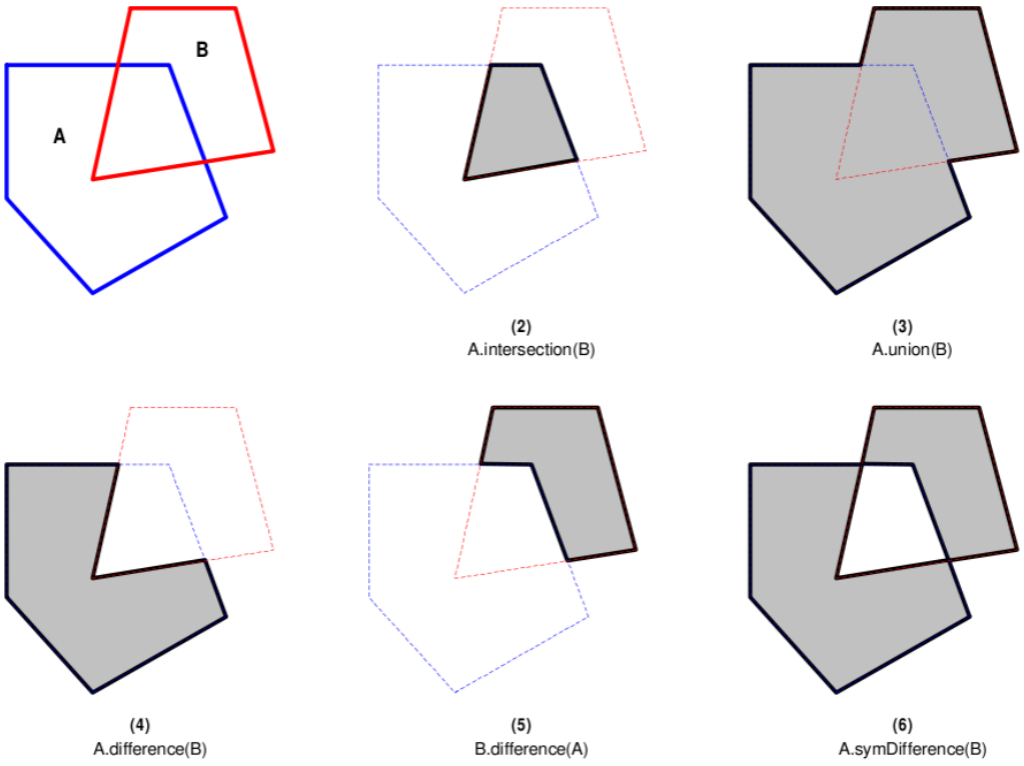


Figura 3.58: Métodos de análisis espacial de JTS.

3.3.8 **Libreria JFreeChart**

JFreeChart es una librería libre 100 % de gráficas desarrollado en Java. Facilita a los desarrolladores mostrar gráficos de calidad profesional en sus aplicaciones.

JFreeChart soporta gráficos circulares (2D y 3D), gráficos de barras (horizontales y verticales, regular y apilado), gráficos de líneas, gráficos de dispersión, gráficos de series de tiempo, diagramas de máximos, mínimos y de apertura y cierre, parcelas, diagramas de Gantt, parcelas combinado, termómetros, relojes, etc. JFreeChart se puede utilizar en las aplicaciones, applets, servlets y JSP. Este proyecto es mantenido por David Gilbert.

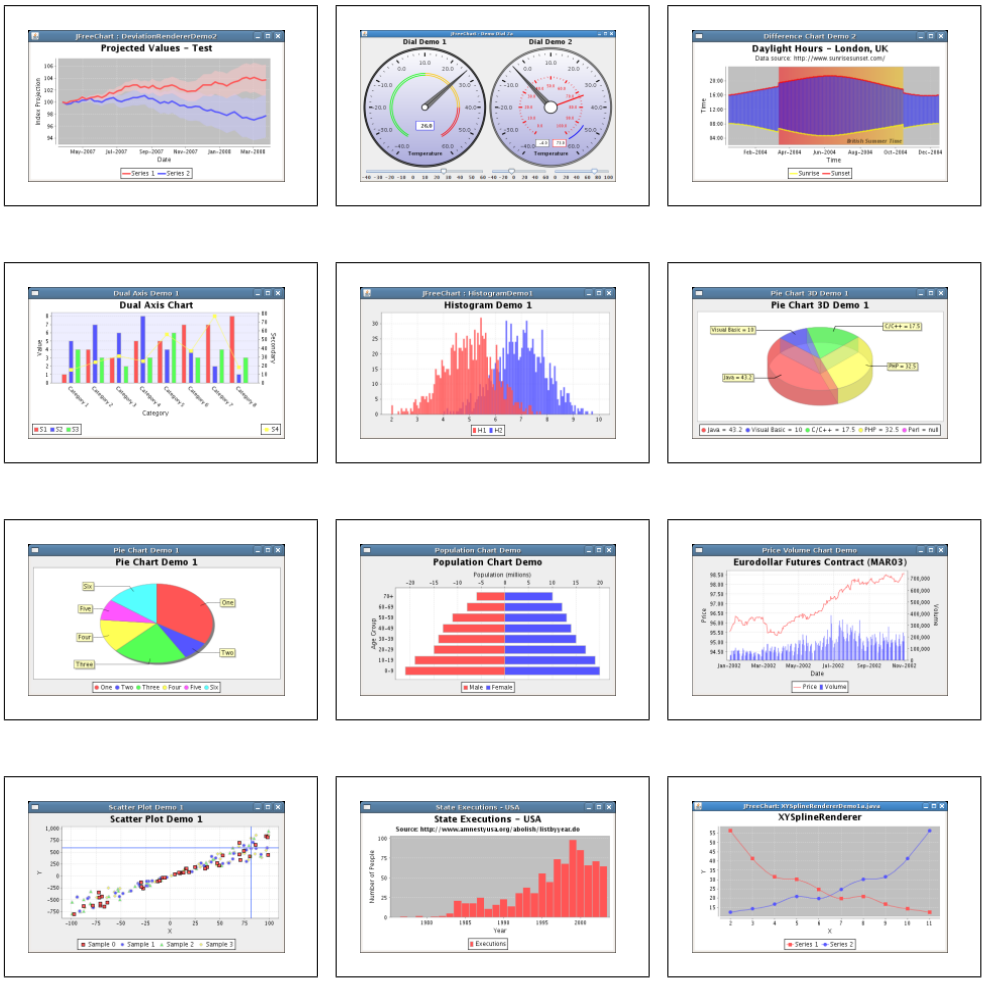


Figura 3.59: Distintos tipos de gráficas de JFreeChart.

JFreeChart incluye:

- Una API bien documentada, que da soporte a una amplia gama de tipos de gráficas.
- Un diseño flexible que es fácil de ampliar, dirigido tanto al lado del servidor como a las aplicaciones del lado del cliente.
- Soporte para tipos de salida, incluyendo los componentes Swing, archivos de imagen (incluyendo PNG y JPEG) y gráficos vectoriales (incluyendo PDF, EPS y SVG).
- JFreeChart es software libre. Se distribuye bajo licencia GNU Lesser General Public Licence (LGPL), que permite su uso en aplicaciones propietarias.

3.4 Software LiDAR

En este apartado se va a analizar una completa herramienta comercial que trabaja con datos LiDAR. Se trata de una serie de paquetes que cubren diferentes propósitos en el proceso de utilización de datos LiDAR. Estos paquetes son combinables entre sí de modo que es posible disponer de la herramienta completa que permita ajustar los datos tras la adquisición, su procesado eficiente y su utilización para diferentes tareas habituales.

3.4.1 Productos de Terrasolid

Los paquetes software Terrasolid se utilizan en todo el mundo para los datos procedentes de Lidar aerotransportado y móvil. Cada paquete cumple requisitos de su tarea específica. Sin embargo es posible la agrupación de diferentes aplicaciones para la realización de un trabajo completo: calibración de los datos, la clasificación de puntos, procesado de imágenes y la creación de productos finales en un entorno de software.

Terrasolid dispone de varias categorias de productos:

- **Procesado de datos LiDAR**
- **Utilización de datos LiDAR**
- **Diseño de infraestructuras**
- **Redes de tuberias**

Terrasolid es una suite de software estándar para el tratamiento de datos láser aerotransportado y móvil y de imágenes. *Un flujo de trabajo centrado en el procesamiento es el factor clave para alcanzar la mayor productividad y mejor calidad con el menor esfuerzo* aseguran en la compañia.

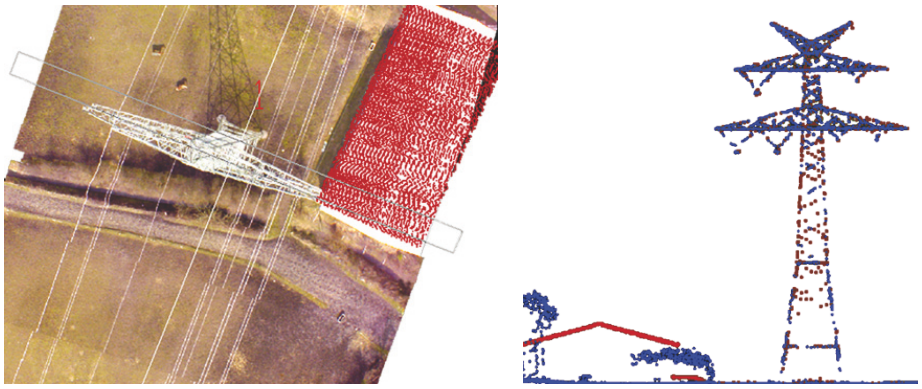
Terrasolid ofrece la posibilidad de adquirir una combinación que se adecue desde el punto de vista del proyecto y los productos de entrega solicitados. Al iniciar un proyecto se cuenta con puntos de láser en bruto e imágenes, trayectorias y la calibración de la cámara una selección de paquetes apropiada puede ser un conjunto de TerraScan, TerraModeler, TerraPhoto y TerraMatch. La calibración de datos antes de la clasificación del punto final es casi sin excepción una obligación. TerraMatch es el paquete para este propósito. TerraPhoto es una combinación natural de TerraScan, TerraModeler y TerraMatch. Los datos láser y el procesamiento de imágenes se apoyan entre sí. Las imágenes permiten una forma sencilla de controlar y cambiar la ubicación en xy de los puntos láser.

La suite de aplicaciones incluye **paquetes** que se detallan en los siguientes subapartadaos.

3.4.1.1 TerraScan

TerraScan es el paquete de Terrasolid para la lectura de los datos de los puntos y su clasificación en diferentes clases. Con TerraScan se puede procesar datos de láser aerotransportado y móvil de proyectos de cartografía. La cantidad total habitual de datos de puntos láser es enorme. Al leer todos los puntos al mismo tiempo para su procesado fácilmente se exceden los límites de memoria del sistema operativo Windows. Para evitar estos obstáculos TerraScan permite dividir los conjuntos de datos en bloques geográficos más pequeños y luego automatizar su procesado por bloques. Cada licencia de TerraScan va

acompañada de una licencia de TerraSlave. TerraSlave libera su trabajo con MicroStation para hacer cualquier otro trabajo.



(a) Imagen LiDAR en xy (b) Corte longitudinal

Figura 3.60: Visualización de un corte longitudinal de datos LiDAR

Las principales **características** de TerraScan son las siguientes:

- Leer los puntos láser en bruto como archivos de texto o archivos binarios XYZ como LAS y TerraScan binario.
- Visualizar los puntos en tres dimensiones.
- Definir sus propias clases de puntos, tales como suelo, vegetación y edificios.
- Clasificar los puntos en el suelo, la vegetación, edificios, etc.
- Dividir los puntos en bloques.
- Automatizar procesamiento de macros.
- Clasificar puntos usando algoritmos automáticos.
- Eliminar los puntos innecesarios o erróneos en un área determinada.
- Digitalizar entidades haciendo snapping[10] a los puntos láser.
- Detectar los cables de alimentación de línea y los techos de edificios.

[10]La herramienta de snapping o anclaje asiste en multitud de herramientas CAD haciendo que al acercar el cursor sobre otra entidad éste se coloque justo encima de ella.

- Exportación de elevación de puntos como imágenes raster a color.
- Proyectar puntos en perfiles longitudinales.
- Salida clasificada puntos como archivos de texto.

Gracias a estas características TerraScan puede:

- Clasificar suelo y la vegetación por separado.
- Detección de catenarias realizando los mínimos ajustes.
- Calcular las ecuaciones de la catenaria.
- Pasar los puntos de suelo a TerraModeler para el modelado de superficies.
- Producción de informes de datos computados.

3.4.1.2 TerraSlave

Con TerraSlave es posible compartir las tareas de procesamiento entre los ordenadores a través de LAN. TerraScan, TerraPhoto or TerraMatch se encargan de dividir los datos en bloques. A continuación, distribuye los datos y las tareas a través de LAN para el procesamiento por lotes por TerraSlave.

Un sistema de escaneado láser aerotransportado produce 30,000 a 150,000 puntos por segundo. El tamaño de un proyecto de estudio varía desde 10 millones hasta 50 billones de puntos. Además de los datos por láser, a los conjuntos de datos se incluyen a menudo las imágenes digitales capturadas durante la toma de datos. Las imágenes ocupan espacio en disco que más puntos láser. En un proyecto típico de un billón de puntos de láser el tamaño de los puntos del láser es de 24 GB y 120 GB de imágenes. Estas masas de datos son un verdadero desafío para la capacidad de procesamiento de cualquier PC. TerraSlave es una solución optimizada para compartir las tareas de procesamiento entre los ordenadores particulares o servidores de red.

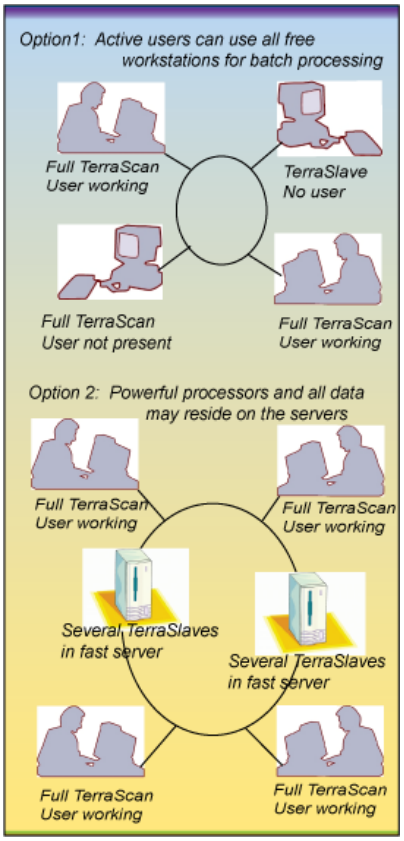


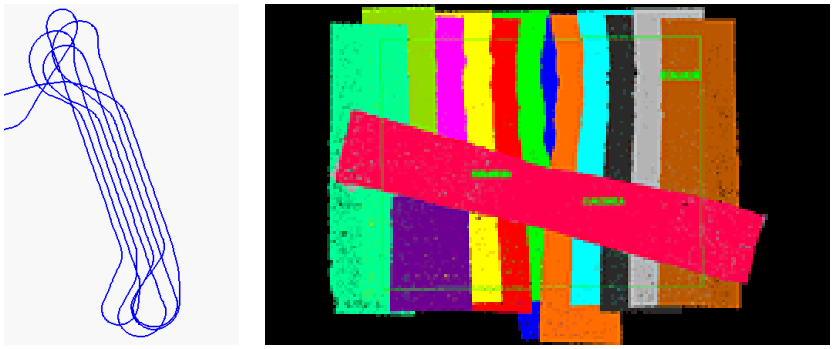
Figura 3.61: Esquema de TerraSlave

3.4.1.3 **TerraMach**

Permite la corrección automática de la posición y los parámetros de orientación para proporcionar una mayor precisión en los productos finales. En este proceso TerraMatch compara el solape de las diferente líneas de vuelo entre sí. Durante el proceso mide las diferencias de tiempos de retorno o la intensidad de los puntos de láser de las diferentes pasadas del láser. Después busca el mejor método de ajuste de los parámetros de orientación. El usuario puede decidir si TerraMatch ajusta todos los datos o bien sólo los puntos de las trayectorias seleccionadas.

Las **características** principales del software son:

- Un procedimiento completamente automático para la corrección de las superficies escaneadas con láser.
- Modelos rígidos de errores en la orientación.



(a) Línea de vuelo (b) Pasadas del láser

Figura 3.62: La línea de vuelo y diferentes pasadas del láser útiles para calibrar los datos.

- Ajuste por mínimos cuadrados[11] para la estimación de los errores de orientación.
- Las observaciones son las diferencias en la elevación y / o la intensidad.
- Zona de juego basado en adaptarse a la geometría de escaneo láser.
- Datos para la detección de errores graves.

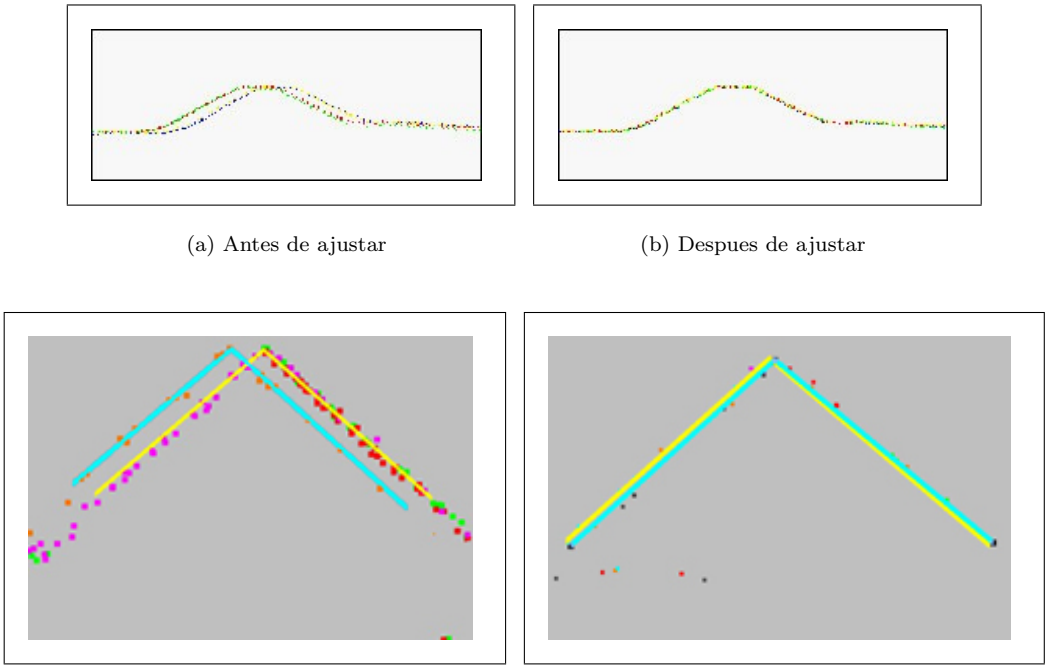
Las **ventajas** principales del software son:

- Ajuste automático y control de datos láser.
- Integrado con TerraScan, TerraModeler y TerraPhoto para procesamiento de datos láser.

3.4.1.4 TerraModeler

TerraModeler es el paquete de Terrasolid para la validación de los puntos del láser y la creación de modelos de superficie de puntos láser.

[11] El método de los mínimos cuadrados asume que la curva de mejor ajuste de un determinado tipo es la curva que tiene la suma mínima de las desviaciones cuadradas (error de los mínimos cuadrados) de un determinado conjunto de datos.



(a) Antes de ajustar

(b) Despues de ajustar

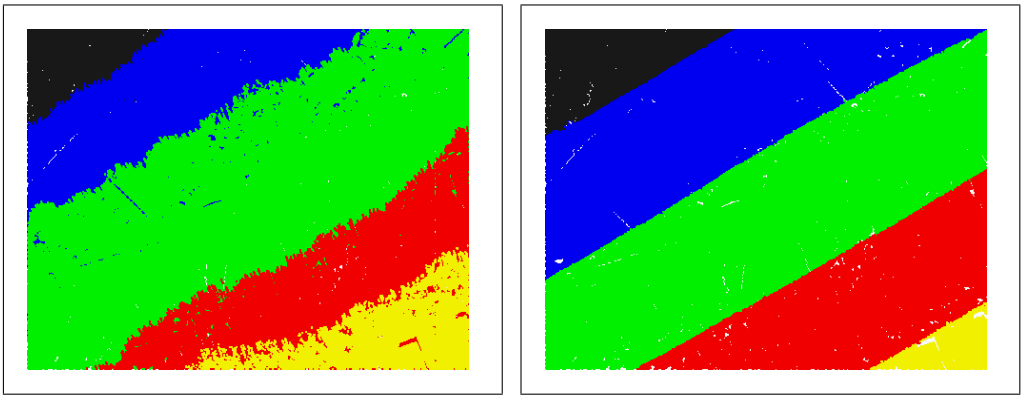
(c) Tejado antes de ajustar

(d) Tejado después de ajustar

Figura 3.63: Datos antes y después de ser ajustados.

TerraModeler trata la creación, edición y uso de superficies del terreno. Crea modelos digitales del terreno, capas de la tierra o diseña elementos mediante la lectura en los puntos láser, elementos de diseño gráfico o archivos de texto XYZ. Funciones versátiles para editar y mostrar modelos digitales del terreno, curvas de nivel, perfiles, así como calcular volúmenes entre modelos digitales de superficie permiten que TerraModeler se utilice con propósitos medioambientales. Puede visualizar modelos de superficies por curvas de nivel, malla, números, superficie sombreada,etc. La vista de superficie sombreada permite una visualización interactiva para identificar y validar la clasificación de puntos láser. La pantalla se actualiza inmediatamente para reflejar los cambios realizados. TerraModeler mantiene abiertos modelos de superficie en la memoria RAM del ordenador. Esta característica le permite utilizar un gran número de modelado de puntos y distintos modelos del terreno vinculados al mismo archivo de diseño durante la misma sesion. TerraModeler es una excelente opción para:

- Ingenieros civiles y geotécnicos.



(a) Antes de eliminar solape (b) Después de eliminar solape

Figura 3.64: Datos LiDAR antes y después de eliminar el solape entre distintas pasadas del avión.

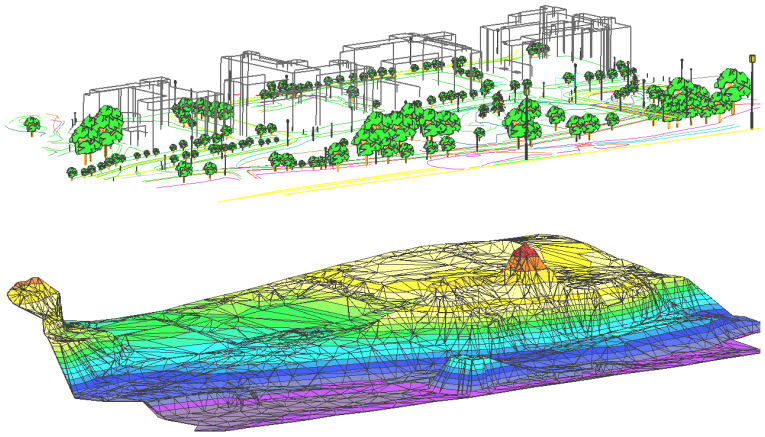


Figura 3.65: Modelos digitales de TerraModeler.

- Arquitectos y urbanistas.
- Cartógrafos en tareas de fotogrametría.
- Ingenieros medioambientales
- Los contratistas para el cálculo de masas entre superficies

3.4.1.5 TerraPhoto

TerraPhoto es el paquete de Terrasolid para la transformación y rectificación de las imágenes aéreas para ortofotos mediante el uso de puntos láser del suelo como superficie de proyección. TerraPhoto

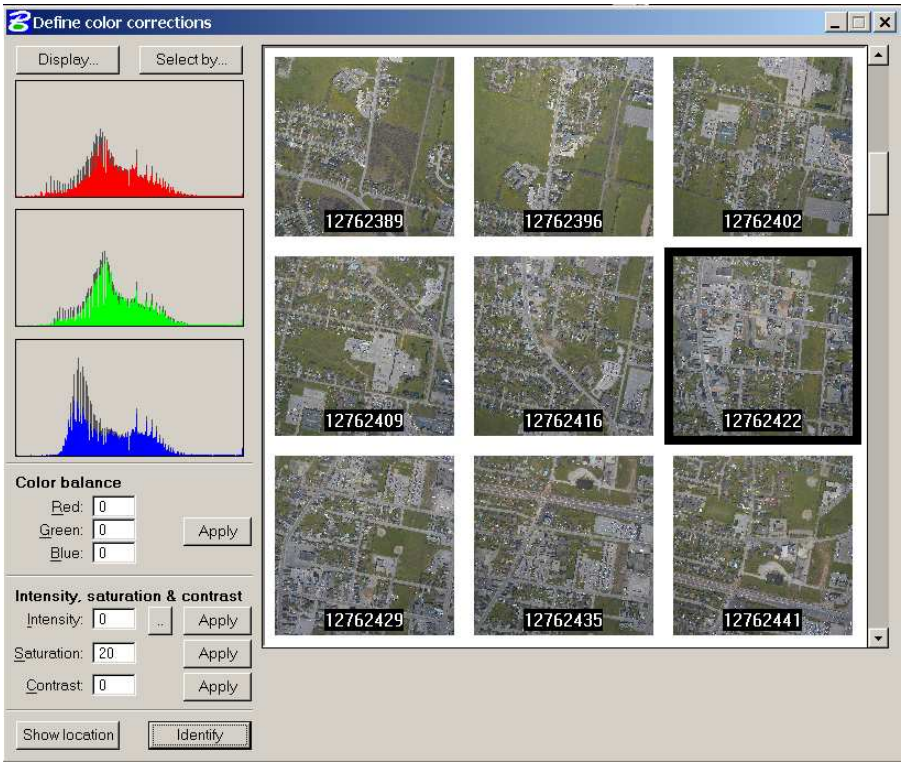


Figura 3.66: Corrección de color de ortofotos.

está específicamente preparado para el manejo de imágenes digitales tomadas por las cámaras durante una misión de exploración láser. Cualquier foto aérea puede ser usada como fuente de datos, si no hay muestras digitales disponibles. Como entrada TerraPhoto necesita imágenes RAW, trayectorias, puntos láser de suelo clasificados y el archivo de calibración de laboratorio de la cámara. Las etiquetas de tiempo sincronizan las imágenes con las trayectorias. Para alcanzar la máxima precisión de la calibración de la cámara se realizará siempre mediante el uso de los puntos de enlace definido por TerraPhoto. El proceso de ortorectificación completa se puede realizar sin tener puntos conocidos en el sitio. El método de rectificación tiene las siguientes ventajas:

- En una sola pasada crea directamente un mosaico de rasters ortorrectificados.
- El modelo de superficie triangulada láser sigue exactamente todas las características del terreno. El sistema calcula un valor de elevación para cada píxel en la imagen orto.

- Suavizado automático de las transiciones de color entre las imágenes.

Un combinación de TerraModeler y TerraPhoto permite cubrir modelos digitales del terreno con ortofotos e ilustrar su modelo. Para la exhibición final es posible crear escenas y animaciones. Los formatos de archivo soportados incluyen ECW, GeoTIFF, TIFF, BMP, CIT, COT, RLE, PIC, PCX, GIF, PNG y JPG2000.

3.4.1.6 **TerraSurvey**

TerraSurvey es el paquete de Terrasolid para leer datos de estaciones totales y GPS tomados en campo, perfiles de visualización como parte del control de calidad de los datos Lidar. TerraSurvey lee datos de la toma de datos como archivos de texto y crea un archivo de un estudio de diseño en 3D. Reconoce de forma automática una serie formatos del estudio de Trimble Leica, etc. Al permitir definir su propio formato de archivo se puede leer prácticamente cualquier archivo basado en los campos de coordenadas o ángulos. A cada punto del estudio se le asigna un código, que define al objeto del estudio. La representación gráfica de cada código se define por una o varias normas de dibujo. El usuario puede crear nuevos códigos y normas de dibujo o modificar los ya existentes y, finalmente, guardarlas como librería de código. TerraSurvey y TerraModeler combinan para la creación de modelos digitales del terreno automáticos a partir de elementos del estudio. Por lo tanto cada código tiene un sub-código que indica cómo TerraModeler va a usar los puntos del estudio en el modelado del terreno. Cuando TerraModeler escanea los elementos de mapeo codificados por TerraSurvey, reconoce automáticamente el modelado de información y decide si el punto es un punto aleatorio, línea de rotura, o el punto no pertenece a ningún modelo. Utilizar TerraSurvey con TerraPhoto TerraScan permite calcular el mejor desplazamiento para ortoimagen mosaico para responder la ubicación de las señales de tierra estudiadas. También puede dar códigos a los objetos vectorizados como cables eléctricos y líneas de caminos, que hayan sido detectados por TerraScan. Esto puede ser muy útil, cuando se trata de exportar los datos a bases de datos y otros sistemas de MicroStation.

3.4.1.7 TerraPipeNet

TerraPipeNet es la aplicación de Terrasolid para el agua, aguas residuales y la gestión de la información geográfica y el control de incidencias de las redes. Toda la información de la red como la localización, los atributos de los elementos, informes de incidencias y cualquier evento de mantenimiento se pueden almacenar en TerraPipeNet. El sistema de seguimiento y las capacidades de reportar incidencias hacen que TerraPipeNet garantice una acción rápida para problemas encontrados y una manera para optimizar la renovación de redes.

3.4.1.8 TerraPipe

TerraPipe para el diseño de líneas de agua y alcantarillado tubería. Todo el diseño del drenaje, alcantarillado, agua potable y otras redes de tubería se realiza con elementos en tres dimensiones. El subsuelo en las zonas urbanas está a menudo lleno de toda clase de redes. Por lo tanto el espacio para tender tuberías nuevas es limitada. El abordaje de 3D TerraPipe resulta una forma de comprobar el espacio disponible. TerraPipe muestra e informa de la distancia más corta en la ventana de visualización. Las principales **características** son las siguientes:

- Ajustes configurables de acuerdo a las necesidades locales y las normas.
- Instrucciones para leer en las tomas de datos y generar una red de tuberías correspondientes mediante elementos diseñados en 3D.
- Guarda los datos como archivos binarios o directamente a la base de datos de TerraPipeNet.
- Dispone de herramientas para diseñar, editar o borrar elementos de la red.
- Dispone de herramientas para generar y actualizar mapas, perfiles e informes.
- Metodos de trabajo y funciones para el estudio, editción y creación de modelos en 3D de la red de tuberías existentes.

3.5 Otras aplicaciones relacionadas

3.5.1 SEXTANTE

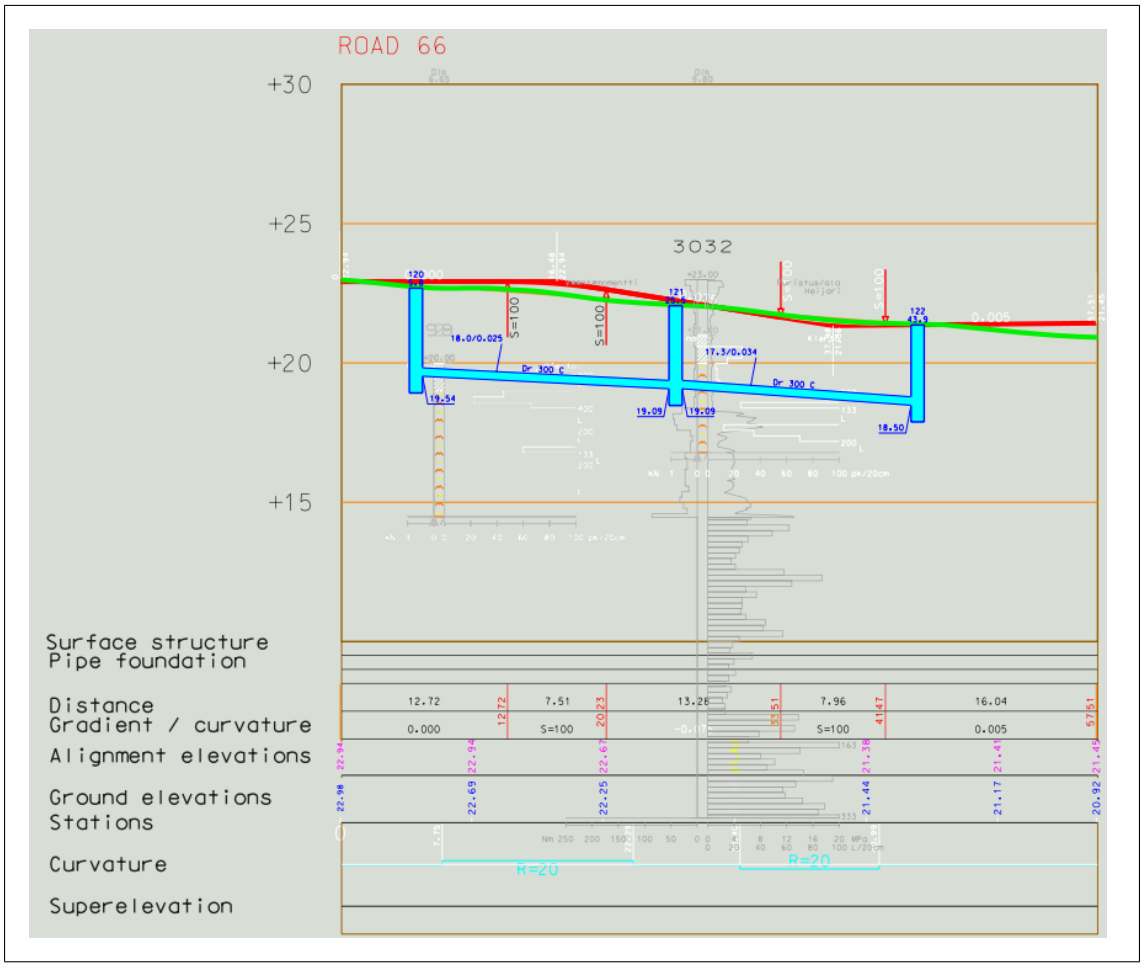


Figura 3.67: Diseño de la red de tuberías.

El proyecto SEXTANTE se inició en 2004 con el objetivo principal de desarrollar una solución SIG diseñado especialmente para las necesidades forestales regionales. A pesar de que estaba destinada originalmente a profesionales de la gestión forestal, ha demostrado ser una solución para cualquier usuario con necesidad de fuertes capacidades de análisis geoespacial, y se desarrolla actualmente como tal. Elementos adicionales para el inventario forestal se están desarrollando.

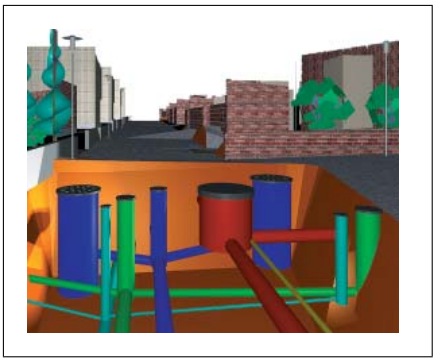


Figura 3.68: Visualización de la red de tuberías en 3D.

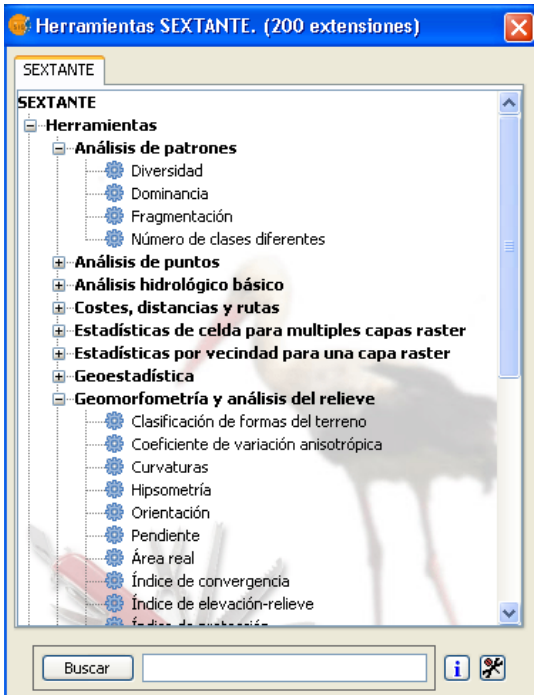


Figura 3.69: Gestor de extensiones de SEXTANTE.

La primera versión de SEXTANTE tuvo como base el software alemán SAGA, un SIG centrado principalmente en el análisis. SAGA originalmente disponía de un conjunto de 120 módulos de análisis que se enriqueció con más de 70 nuevos, y algunas modificaciones se efectuaron también en el núcleo del sistema. Se produjo una relación muy estrecha entre SAGA y los equipos de SEXTANTE, y ambas extensiones y la modificación del nucleo finalmente se dirigieron a la distribución SAGA oficial y están hoy en día incluidos en los lanzamientos actuales de SAGA.

En ese momento, gvSIG no era aún un producto maduro SIG, y se

consideró inadecuado para los objetivos del proyecto. Sin embargo, pronto gvSIG experimentó un crecimiento impresionante y se convirtió rápidamente en un SIG hecho y derecho, que incluía características que no se encuentraban en SAGA, como los servicios de conexiones web. La decisión fue tomada para migrar todos los trabajos anteriores y aplicar todos los conocimientos adquiridos en el trabajo con SAGA a gvSIG para convertirlo en una herramienta de análisis geoespacial poderosa. Aunque rico en funcionalidades, en gvSIG había una falta de funciones de análisis (a excepción de un pequeño conjunto de geoprocesos de capas vectoriales, incluyendo operaciones como el almacenamiento en búfer, cortar, unir una combinación, entre otros), por lo que el resultado sería beneficioso para ambas partes.

Se dieron los siguientes pasos para desarrollar la versión para gvSIG de SEXTANTE:

- Creación de una capa base sobre la que las extensiones de análisis geoespacial podrían ser implementadas fácilmente. Eso permite encapsular la complejidad de la extensión de gvSIG y la arquitectura plugin, y hacer más fácil la implementación de un geoalgoritmo nuevo, siguiendo las ideas de SAGA.

- Migración de todas las extensiones originales de Saga y todas las desarrolladas en la versión anterior de SEXTANTE para gvSIG, usando la capa base antes mencionada. Algunas extensiónes no relacionadas con el análisis, como de entrada/salida, no se han aplicado, puesto que ya existía en gvSIG. Otras nuevas se han añadido también, hasta un total de más de 220, aproximadamente la mitad de las cuales provienen de la versión original de SAGA.

- Incluye elementos novedosos que permiten aprovechar mejor las posibilidades del conjunto de extensiones de análisis.

En la actualidad, SEXTANTE no es un conjunto de extensiones para gvSIG, sino una biblioteca independiente de Java basado en el código desarrollado para esta versión de gvSIG anterior. De esta manera, puede ser incorporado fácilmente en gvSIG para agregar las mismas

funcionalidades de la versión anterior, pero también en otras aplicaciones SIG. Esto incluye otros SIG de escritorio, otros tipos de aplicaciones y librerías, o incluso para servir geoalgorithms SEXTANTE a través de WPS[12].

SEXTANTE desarrollado por la Universidad de Extremadura para la Junta de Extremadura , a través de la Titulación de Ingeniería Forestal del Centro Universitario de Plasencia para satisfacer necesidades, especialmente en el ámbito forestal.

SEXTANTE tiene una amplia gama de usuarios, desde usuarios ocasionales o habituales de de SIG, hasta usuarios sin experiencia al respecto. Su estructura modular y su enfoque analítico hacen que sea especialmente adecuado para su uso en el ámbito académico y de investigación ya que la aplicación se estructura de forma que la implementación de algoritmos de análisis espacial sea sencilla. Maneja y analiza información tanto vectorial como ráster aunque con un especial enfoque hacia esta última, pues es en el manejo de este tipo de datos donde reside la verdadera potencia del programa.

SEXTANTE distribuido bajo licencia GPL hace posible que algunos usuarios con conocimientos de programación desarrollen nuevos algoritmos que resuelvan problemas no contemplados por los módulos distribuidos con el programa.

SEXTANTE dispone de los siguientes elemntos básicos:

- **Gestor de extensiones**. Se trata del elemento principal de SEXTANTE. En él se encuentran el conjunto de todas las extensiones que pueden ejecutarse de forma individual (ver imagen 3.69). Las extensiones aparecen agrupadas en bloques de acuerdo con el tipo de análisis que llevan a cabo. En la parte inferior encontramos el botón Buscar mediante el que es posible filtrar las extensiones para localizar la deseada. En él aparecen en color negro o gris las

[12] OpenGIS Web Processing Service (WPS) Interfaz estándar que proporciona reglas para estandarizar la forma en entradas y salidas para servicios de procesamiento geoespacial, tales como la superposición de polígonos. Define cómo un cliente puede solicitar la ejecución de un proceso, y cómo el resultado del proceso se maneja. Define una interfaz que facilita la publicación de procesos geoespaciales y el descubrimiento de los clientes y el enlace a esos procesos. Los datos requeridos por el WPS se pueden entregar a través de la red o pueden estar disponibles en el servidor.

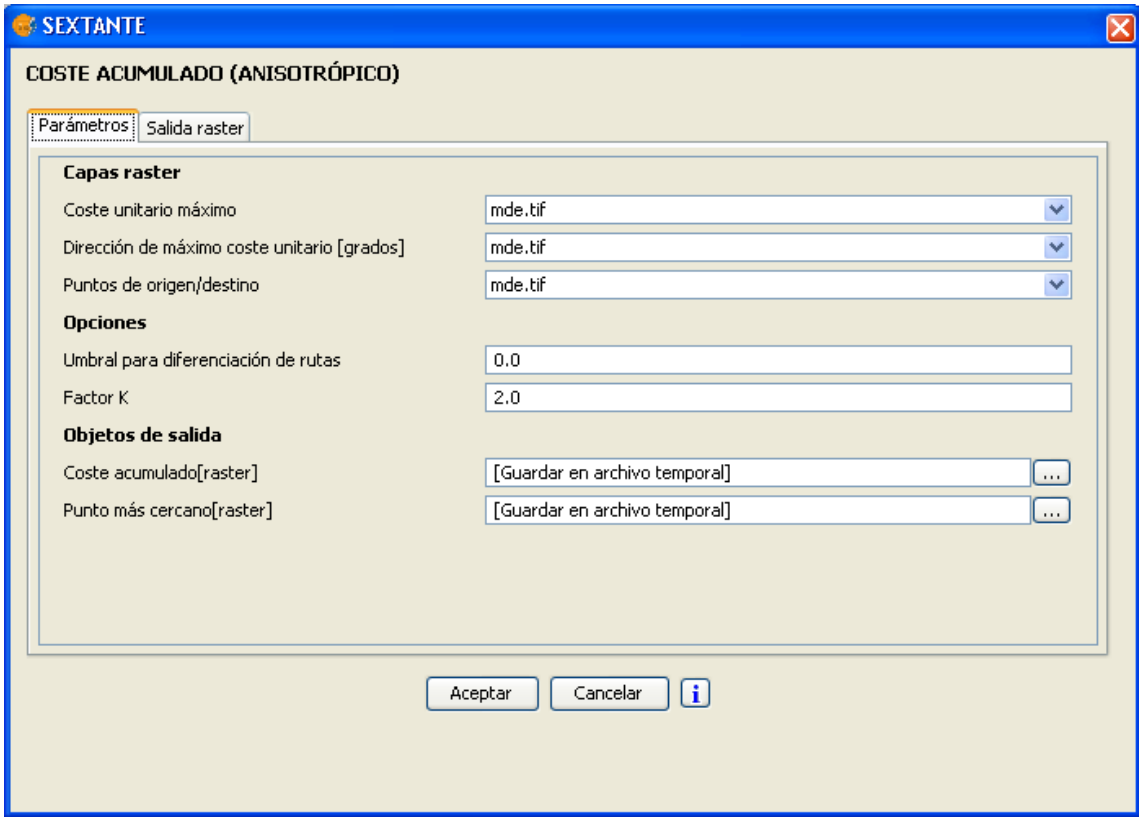


Figura 3.70: Extensión de SEXTANTE.

extensiones en función de si la información disponible en la ventana activa es suficiente para poder ejecutar la extensión o no. Una vez ejecutada una extensión realizando doble click sobre ella aparece una nueva pantalla de recogida de los parámetros de entrada (ver imagen 3.70) y en algunos casos con la posibilidad de crear nuevas capas de salida.

- **Modelizador gráfico.** Esta herramienta permite la creación de complejos modelos mediante una sencilla interfaz. De esta forma simplifica procesos que impliquen el uso de varias extensiones de forma encadenada. Así poder realizar una extensión que tome datos del usuario como parámetros de entrada y ejecute extensiones cuya salidas alimenten la entrada de nuevas extensiones encadenadas para conformar la extensión general.

- **Proceso por lotes.** Ls extensiones de SEXTANTE, incluyendo los modelos, pueden ser ejecutadas como *proceso por lotes*, es decir, pueden ser ejecutadas repetidamente sobre un conjunto de parámetros de entrada sin tener que realizar varias ocasiones lla-

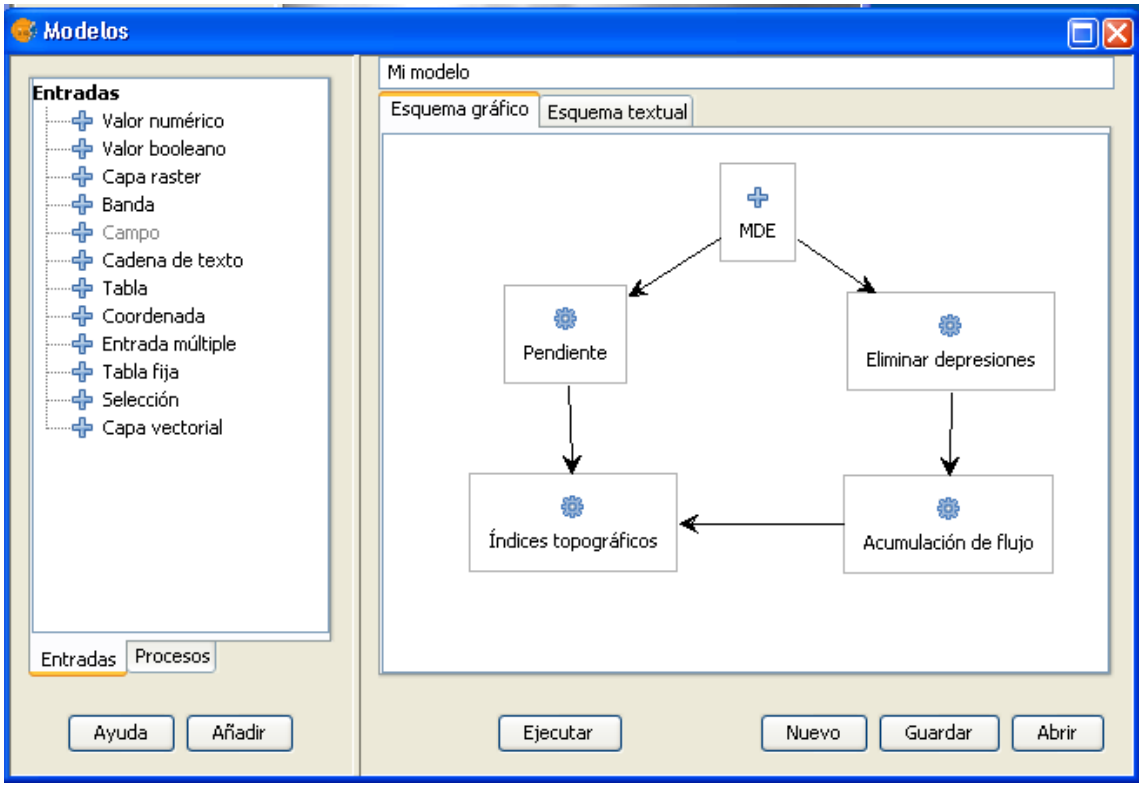


Figura 3.71: Modelador gráfico de SEXTANTE.

Figura 3.72: Ejecución por lotes de SEXTANTE.

madas a la extensión desde el gestor de extensiones. De este modo es posible, por ejemplo, la realización de una operación como puede ser un filtro sobre una serie de capas.

- **Línea de comandos.** Mediante la línea de comandos los usuarios avanzados de SEXTANTE pueden automatizar taeas creando sencillos scripts.

La web oficial del proyecto es http://www.sextantegis.com. Algunas de las utilidades disponibles en SEXTANTE:

- Análisis de patrones.
- Análisis hidrológico básico.
- Costes, distancias y rutas.
- Estadísticas de celda para múltiples capas raster.
- Estadísticas por vecindad para una capa raster.

Figura 3.73: Línea de comandos de SEXTANTE.

- Geoestadística.
- Geomorfometría y análisis del relieve.
- Herramientas básicas para capas raster.
- Herramientas de análisis para capas raster.
- Herramientas de cálculo para capas raster.
- Herramientas para capas de líneas.
- Herramientas para capas de puntos.
- Herramientas para capas de polígonos.
- Herramientas para capas raster categóricas.
- Herramientas para capas discretas e información categórica.
- Herramientas para capas vectoriales.
- Herramientas para crear nuevas capas raster.
- Herramientas para tablas.
- Iluminación y visibilidad.
- Localización óptima de elementos.
- Lógica difusa.
- Métodos estadísticos.
- Perfiles.
- Rasterización e interpolación.
- Tratamiento y análisis de imágenes.

- Vectorización.
- Zonas de influencia (buffers).
- Índices de vegetación.
- Índices y otros parámetros hidrológicos.

CAPÍTULO 4

Planificación y evaluación de costes

4.1 Consideraciones previas

En el proceso de gestión para la creación de un sistema o software, una de las actividades es la planificación. La planificación consiste en una estimación en la utilización de ciertos recursos como tiempo, coste económico y esfuerzo. Estimar es calcular el valor aproximado de una cosa. Aunque la estimación, es más un arte que una ciencia exacta, es una actividad importante que no debe llevarse a cabo de forma descuidada.

Existen técnicas útiles para la estimación de costes de tiempo. Dado que la estimación es la base de todas las demás actividades de planificación del proyecto sirve como guía para una buena Ingeniería de software. Al estimar tenemos en cuenta no solo el procedimiento técnico a utilizar en el proyecto, sino que se tienen en cuenta además los recursos, costos y planificación.

El tamaño del proyecto es otro factor importante que puede afectar la precisión de las estimaciones. A medida que el tamaño aumenta,

crece rápidamente la interdependencia entre varios elementos del software y del mismo modo crece la incertidumbre a la hora de estimar costes del proyecto.

La experiencia y disponibilidad de información de otros proyectos es otro elemento que determina el riesgo de la estimación.

4.2 Planificación inical

El periodo de prácticas en la empresa Dielmo 3D S.L. por la normativa no podía extenderse más de 6 meses. Por este motivo en las primeras tomas de contacto con José Carlos García, director gerente de la empresa, se definieron los límites de este proyecto. De esta forma se aseguró que en ese periodo de tiempo fuera posible la realización de un proyecto final de carrera completo e interesante para ambas partes.

En un primer momento no se tenía clara cuál sería la mejor manera de abordar el desarrollo. Lo que sí que estaban definidos eran los objetivos y el sistema de información geográfica sobre el que trabajar.

En ese momento Dielmo ya disponía de un desarrollo previo de un driver de lectura/escritura de datos LiDAR en gvSIG. El siguiente paso consistía en crear herramientas y la infraestructura software necesaria para abordar un proyecto de datos LiDAR. Las primeras reuniones con Jose Carlos y parte del equipo técnico de Dielmo sirvieron para definir el proyecto y cuantificar los objetivos que se pretendían alcanzar.

En el proceso de estimación se distiguieron varias etapas:

- Identificar los núcleos del proceso
- Desglosar los núcleos en actividades
- Cuantificar las actividades
- Relacionar las tareas
- Identificar recursos

- Asignar recursos a las tareas

Como era de esperar la planificación inicial fue imprecisa. Careciendo de experiencia tanto en proyectos de esta naturaleza como de esta embergadura, se era consciente de la inexactitud a la hora de estimar el coste temporal que llevaría el camino hacia los objetivos marcados.

Una serie de hechos que se enumeran a continuación fueron causa de las variaciones entre las estimaciones y los costes reales:

- Dificultades de identificación del problema. Por ejemplo a la hora de estructurar un proyecto LiDAR en gvSIG no se tenía clara la estrategia a seguir. Se empleó cierto tiempo en decidirse por un nuevo documento en gvSIG.
- Problemas con el workspace[1] de gvSIG en eclipse. El sistema gvSIG es modular. Es decir, el código fuente está compuesto por distintos proyectos de extensiones, librerías, y demás proyectos que han de ser montados en un espacio de trabajo workspace de Eclipse. Problemas como dependencias entre proyectos, configuraciónes iniciales o acceso al svn de gvSIG fueron causantes de retrasos respecto a la planificación inicial.
- Problemas en la implementación de partes del proyecto como el documento de gvSIG, las funciones de persistencia, herramienta de gráficas longitudinales etc.
- Algunas dependencias entre tareas no planificadas.
- Mala estimación de tiempos de actividades como recopilar información e implementación de herramientas de conversión entre otras.
- Imprecisa estimación en la planificación de la realización de esta memoria. Se planificó realizarla al terminar el periodo de prácticas de empresa ignorando en ese momento la futura incorporación al mercado laboral. Este hecho provocó la demora en la elaboración de este documento.

[1] Espacio de trabajo de eclipse donde se incorporan los proyectos que contienen el código fuente.

4.3 Seguimiento

Se ha realizado un seguimiento del proyecto. Éste ha puesto de manifiesto una serie de diferencias entre la planificación inicial y la real. El diagrama de Gannt de tiempos reales que ha conllevado la realización de este proyecto se muestra a continuación (ver imagen 4.1).

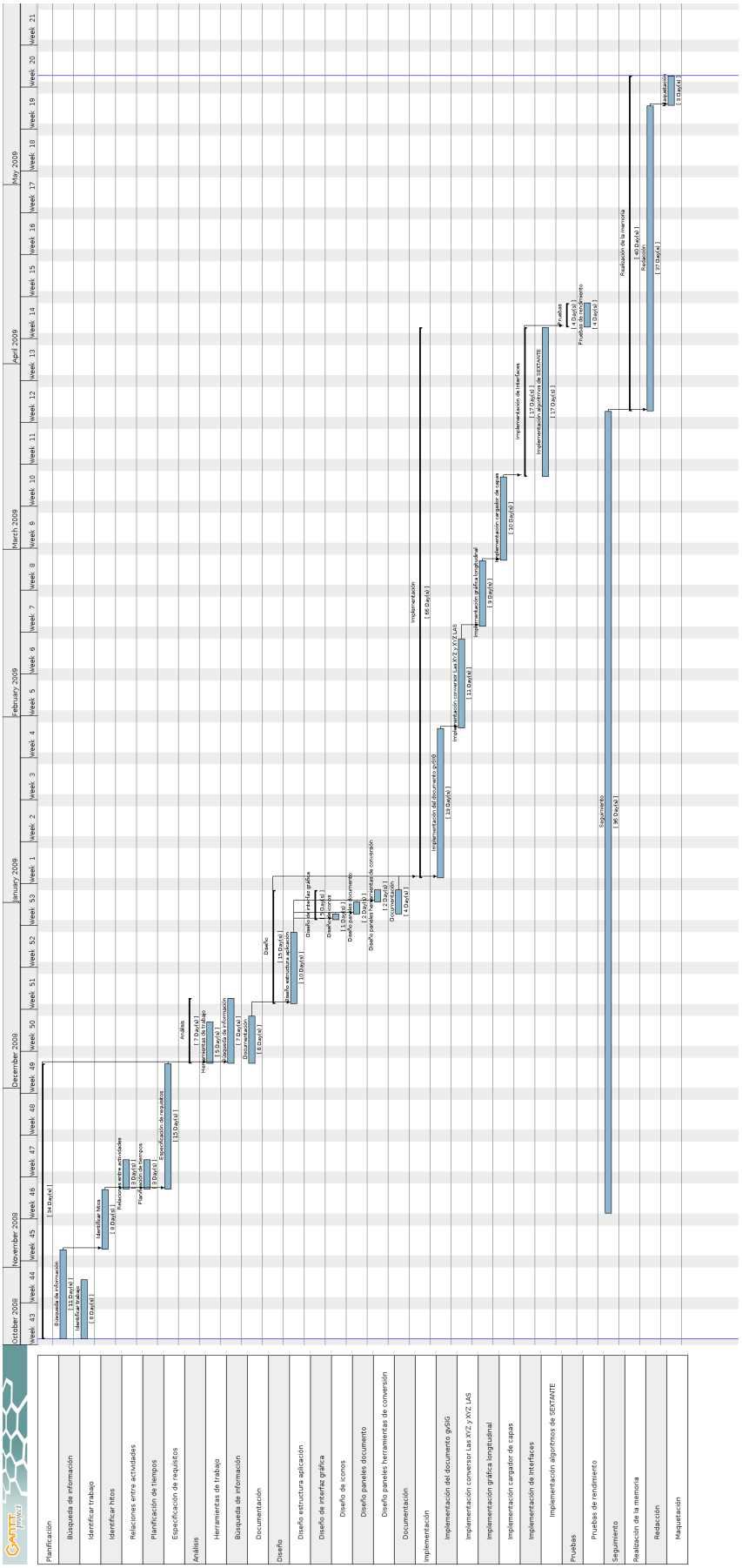


Figura 4.1: Diagrama de Gantt de la planificación del proyecto.

CAPÍTULO 5

Análisis

5.1 Introducción

Son muchas las definiciones existentes para la Ingeniería del Software:

- Según Boehm. La INS supone la aplicación práctica y sistemática del conocimiento científico a la producción de programas que se desarrollan a tiempo y dentro de las estimaciones de presupuesto y la correspondiente documentación para desarrollarlos, instalarlos, usarlos y mantenerlos.
- Según Pressman. La INS es una disciplina que integra métodos, herramientas y procedimientos para el desarrollo de SW de computador.
- Según Davis. La INS es la aplicación de principios científicos para: (1) la transformación ordenada de un problema en una solución SW y (2) el mantenimiento del mismo durante toda su vida útil.
- Según Sommerville. La INS es una disciplina ingenieril que abarca todos los aspectos de la producción de software.

Pese a las distintas definiciones aportadas todos coinciden en la importancia de la disciplina metodológica. De esta manera se minimizan los errores cometidos a lo largo del proceso de desarrollo. Los errores que pudieran cometerse deben ser detectados lo antes posible puesto que errores cometidos en fases tempranas suponen un alto coste de corrección si son corregidos en fases tardías. En las figuras 5.1 y 5.2 se muestran el porcentaje de aparición de errores en las distintas fases del desarrollo de software y el coste que supone la corrección de los mismos según en la fase que se hayan producido respectivamente.

La fase de análisis es la etapa del proceso de desarrollo de sistemas de información que busca obtener la mayor claridad posible con respecto a los requerimientos y necesidades del usuario o usuarios del sistema a desarrollar. La realización de esta fase sirve para la obtenención de documentos de requisitos, modelos relacionales, diagramas de clases, etc. Durante la misma se debe tener especial cuidado, pues de ella depende la calidad y aceptación de los resultados finales. Si un análisis ha sido bien realizado y es muy completo, representa un 70 % de la solución ya obtenida. La fase de análisis o fase inicial para la realización de un desarrollo de software de calidad supone un paso crucial para alcanzar el éxito. La aparición de errores en esta fase y su transmisión a las siguientes fases puede suponer un claro fracaso en la consecución del proyecto.

Entre los factores que determinan la calidad de un software se pueden destacar los siguientes:

- **Corrección** ¿Hace el software lo que se espera de él?
- **Fiabilidad** ¿Lo hace de forma fiable todo el tiempo?
- **Eficiencia** ¿Se ejecutará en el Hw lo mejor que pueda?
- **Integridad** ¿Es seguro?
- **Facilidad de uso** ¿Está diseñado para ser usado?
- **Facilidad de mantenimiento** ¿Puede corregirse?
- **Flexibilidad** ¿Es posible cambiarlo fácilmente?
- **Facilidad de prueba** Es posible probarlo?

- **Reusabilidad** ¿Es posible reusar alguna parte del Sw?
- **Portabilidad** ¿Será posible usarlo en otra máquina o SO?
- **Facilidad de interoperación** ¿Será posible hacerlo interactuar con otro sistema?

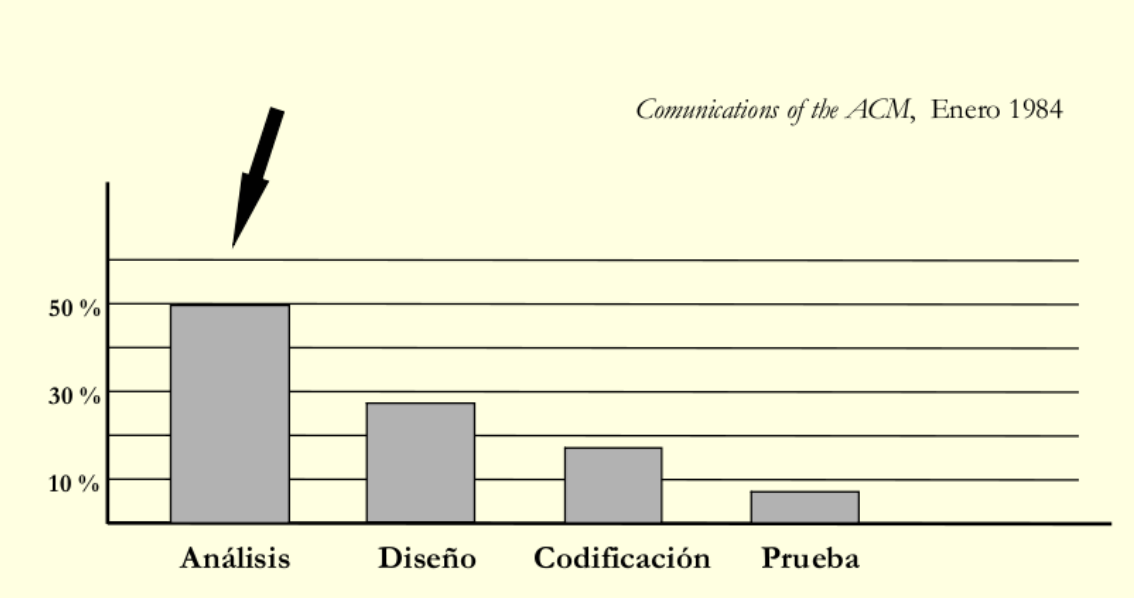


Figura 5.1: Aparición de errores en las distintas fases de desarrollo de software.

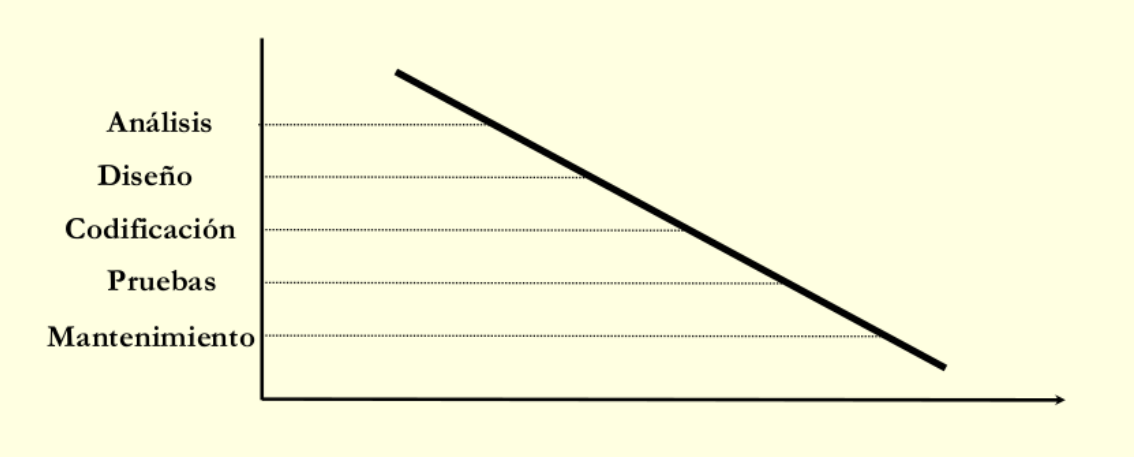


Figura 5.2: Coste de corrección de errores producidos en las distintas fases de desarrollo de software.

5.2 Arquitectura de gvSIG

5.2.1 Introducción

El proyecto gvSIG se presenta como un framework [1] sobre el que se pueden ir añadiendo plugins que le doten de nuevas funcionalidades. La plataforma que proporciona gvSIG se sustenta en una arquitectura abierta. gvSIG usa un modelo que consiste en presentar una serie de herramientas de forma homogénea desde el punto de vista del usuario. Las herramientas se integran dentro de gvSIG usando unos mecanismos llamados plugins.

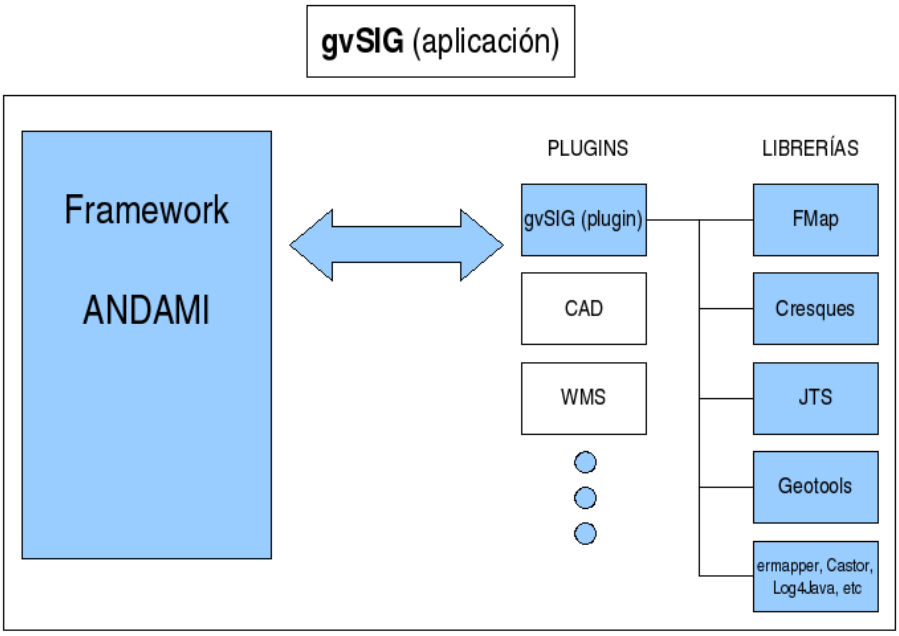


Figura 5.3: Esquema de la arquitectura de gvSIG.

La plataforma gvSIG en si misma está construida a modo de capas, cada una de las cuales define sus propios puntos de extensión. A su vez, cada plugin puede definir sus propios puntos de extensión. Este modelo de plugins, permite a los desarrolladores añadir gran variedad

[1] En el desarrollo de software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (Wikipedia) En el capítulo 5.2.2 se detalla más a fondo el funcionamiento del framework andami.

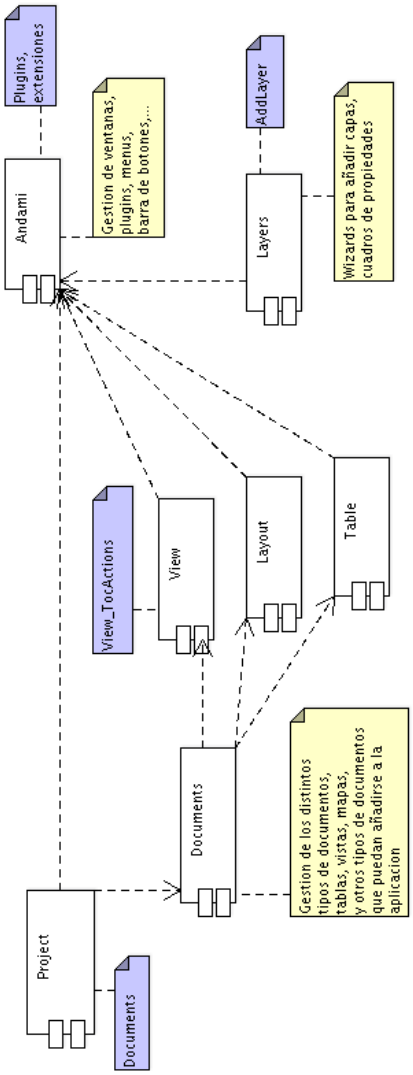


Figura 5.4: Diagrama de subsistemas de gvSIG.

de funcionalidades a la plataforma base de gvSIG, de forma que los artefactos de cada herramienta, como pueden ser los distintos tipos de capas, o botones, se presentan al usuario desde la plataforma común.

Los desarrolladores de plugins también se benefician de esta arquitectura. El framework base de gvSIG les proporciona una serie de servicios de los cuales ellos no tienen que preocuparse, pudiéndose centrar en las tareas específicas de su extensión.

La plataforma gvSIG se sustenta en una arquitectura abierta en la que cada equipo que desarrolla un plugin se puede centrar en su área

de experiencia. gvSIG presenta un conjunto de herramientas de forma homogénea para el usuario. Las herramientas se integran dentro del marco de gvSIG usando unos mecanismos ya definidos llamados plugins. La plataforma gvSIG esta construida a modo de capas, cada una define sus propios puntos de extensión. También cada plugin puede definir sus propios puntos de extensión. Este modelo de plugins, posibilita a los desarrolladores añadir gran variedad de funcionalidades a la plataforma base. Los desarrolladores de plugins se benefician también de esta arquitectura. El framework base de gvSIG les proporciona servicios, pudiéndose centrar en las tareas especificas de su extensión.

gvSIG está estructurada en subsistemas. Éstos están implementados como librerías y como plugins en si mismos. La plataforma gvSIG esta conformada en su núcleo por los siguientes tres subsistemas:

- *gvSIG*. Representa los datos geográficos manejados por Fmap. En este subsistema encontraremos las clases que implementan la mayoría de cuadros de diálogo que utiliza la aplicación, así como las clases de soporte a ésos. Por ejemplo, aquí se encuentran formularios para asignar leyendas, creación de mapas y vistas, definición de escalas, etc.
- *FMap*. Es el corazón SIG de la plataforma. Incluye todas las clases para manejar objetos SIG, como drivers y adaptadores para el manejo de los formatos más usados para el almacenamiento de datos cartográficos. En esta librería encontramos clases de lectura y escritura de los formatos soportados, de dibujo de mapas a las escalas adecuadas, y de asignación leyendas, definición de simbologías, realización de búsquedas, consultas, análisis, etc.
- *Subdriver*. En este subsistema se encuentran las clases que permiten el acceso y la gestión de los datos.

5.2.2 **Andami.**

Un framework es una estructura de soporte, que permite desarrollar una aplicación sobre él. En general, define la arquitectura básica de la aplicación y provee de servicios que agilizarán y simplificarán el desarrollo del proyecto.

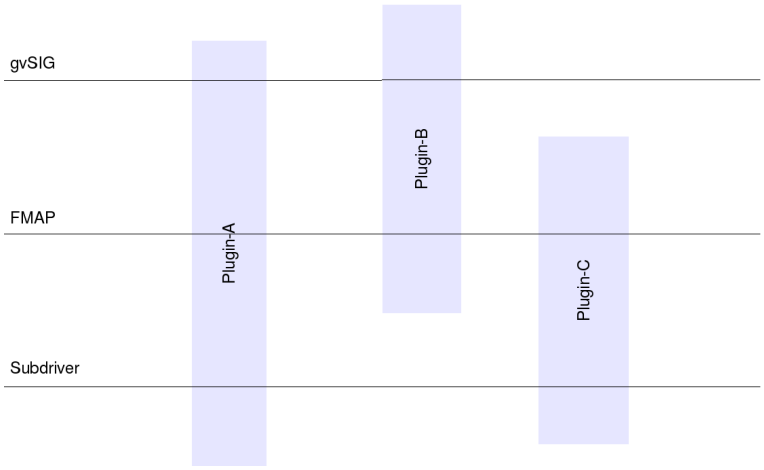


Figura 5.5: Esquema de la arquitectura de gvSIG.

Andami es el framework sobre el que se ha construido gvSIG. Diseñado para ser extensible mediante la implementación de nuevos plugins. Los plugins se consideran módulos que añaden nueva funcionalidad a la aplicación. Andami da soporte, además, a gvSIG en la composición básica de la interfaz de usuario y en la gestión de ventanas y eventos. Proporciona una forma simple importante en la gestión de estos aspectos, así resulta muy sencilla la creación de un plugin funcional de gvSIG.

El proyecto Andami se localiza dentro del código fuente de gvSIG, en el directorio *_fwAndami*. Una vez compilado, genera un fichero andami.jar que se suele situar en la raíz de la estructura de directorios de gvSIG.

Andami contiene la parte de gvSIG encargada de iniciar la ejecución, realiza la carga de plugins e inicializa los subsistemas de la aplicación. Esencialmente, hay dos tipos de entidades que se relacionan con Andami, directa o indirectamente: los plugins y el resto de librerías. Un plugin tiene una estructura especial que Andami reconoce y carga en el arranque, y se conecta con la interfaz de usuario en unos puntos definidos por el plugin. El resto de librerías son aquellas que también son utilizadas en gvSIG pero que no adoptan forma de plugins. La mayor parte de la funcionalidad de gvSIG es proporcionada por los plugins, y éstos a su vez se apoyan en librerías. El propio gvSIG

es un plugin de Andami, que le aporta el concepto de documento (vista, tabla, mapa), capacidad de abrir y guardar proyectos, el gestor de proyectos, y otras funcionalidades. Conviene por tanto distinguir entre referirse al plugin gvSIG (llamado com.iver.cit.gvsig), y la aplicación gvSIG (que engloba Andami, plugin gvSIG, el resto de plugins y librerías).

Las librerías se encuentran en el directorio "*bin/lib*" de la instalación de gvSIG, se incluyen en el CLASSPATH y por tanto están disponibles para toda la aplicación. Las librerías que se encuentran en el directorio de librerías de un plugin sólo están disponibles para éste plugin que las contiene, y para plugins que declaran una dependencia sobre éste. Habitualmente, los plugins tienen definida una dependencia sobre el plugin gvSIG (com.iver.cit.gvsig), de este modo las librerías de gvSIG están disponibles para ellos.

5.2.2.1 **Funcionalidad.**

Andami proporciona la siguiente funcionalidad:

- Extensibilidad con plugins. Carga de plugins, carga dinámica de clases y recursos de los plugins, etc.
- Creación de la interfaz principal desde ficheros XML: menús, barras de herramientas y barra de estado.
- Gestión de ventanas: creación, cierre, propiedades, etc.
- Persistencia de datos: permite a los plugins el almacenamiento de datos en disco y posterior recuperación de forma muy sencilla.
- Traducciones: proporciona servicios de traducción de los textos de la interfaz de usuario.
- Ejecución de tareas en segundo plano: facilita la creación y gestión de tareas que se ejecutan en segundo plano.
- Servicio de registro: aporta un registro (log) de sistema en el que escribir los errores producidos o eventos significativos (útil para detección y depuración de errores).
- Acceso al portapapeles.

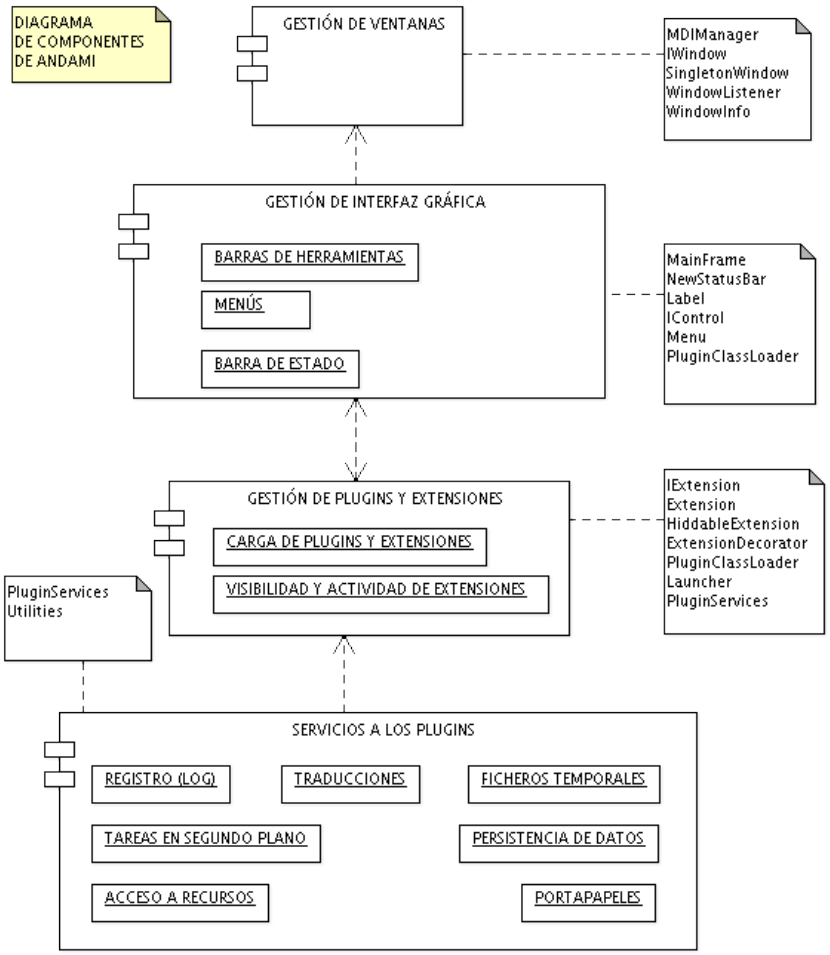


Figura 5.6: Descripción de los bloques funcionales de Andami.

- Creación de ficheros temporales.
- Acceso a recursos de los plugins

Estas funcionalidades que proporciona Andami han sido utilizadas, practicamente en su totalidad, en la realización de este proyecto.

En la figura 5.7 se muestra una perspectiva general de los bloques funcionales de Andami.

- **Gestión de ventanas:**

  Entre la funcionalidad aportada por Andami se encuentra la gestión de ventanas. El programador sólo necesita crear un panel que incorpore el contenido de la ventana, y Andami crea la ventana real.

Andami registra la ventana en el menú de ventanas, y coloca en primer plano la ventana correspondiente al seleccionar una entrada de ese menú. Asimismo Andami posibilita cambiar las propiedades de una ventana (redimensionados, maximizar y minimizar, etc), la obtención de la ventana activa, la lista de ventanas, etc.

- **Gestión de la interfaz gráfica de usuario:**

  Otra funcionalidad que aporta Andami es la creación de la interfaz de usuario y la gestión de los eventos que se produzcan en ella. En el arranque Andami creará la ventana principal, menús, barras de herramientas, barra de estado (con sus controles correspondientes), etc. También, ofrece mecanismos para modificar estos elementos durante la ejecución del programa. La gestión de la interfaz está relacionada la gestión de plugins y extensiones porque la creación inicial de la interfaz se realiza en función de lo que se especifique en los ficheros de configuración de los plugins. Inversamente, las acciones del usuario sobre la interfaz (pulsar un menú o un botón, por ejemplo) tienen efecto también en la gestión de extensiones pues Andami obtendrá la extensión asociada al botón o menú y le notificará la acción que se ha realizado. De esta manera la extensión realizará el procesamiento adecuado para dicha acción. La gestión de ventanas también guarda relación con la gestión de la interfaz ya que las herramientas seleccionables están asociadas a la ventana activa, y por lo tanto cada vez que se cambia de ventana, cambia también la herramienta seleccionada.

- **Gestión de plugins y extensiones:**

  Andami ha sido diseñado para ser extensible mediante plugins y extensiones. Las extensiones son clases Java que añaden alguna funcionalidad a la aplicación, y los plugins son agrupaciones de extensiones que permiten a Andami reconocer y cargar las extensiones. En el arranque, Andami revisa los plugins instalados, carga e inicializa las extensiones que contienen y carga los elementos de la interfaz de usuario que se especifican en los ficheros de con-

figuración del plugin. Cada elemento (botones, menús, controles) está asociado a una extensión concreta y la visibilidad de estos elementos está determinada por dicha extensión. Cuando se pulsa sobre un elemento, se envia una notificación a la extensión asociada.

- **Servicios a los plugins:**

Este último bloque funcional engloba diversos servicios que Andami proporciona a los Plugins. Algunos servicios están ligados a cada plugin y otros funcionan de la misma forma para todos ellos, como se explica a continuación:

- *Servicio de traducciones*: Los plugins tienen unos ficheros de traducciones. Éstos son leídos durante el inicio de la aplicación. Estos ficheros contienen claves de traducción, y una traducción asociada a cada clave, para cada idioma disponible. De este modo, los plugins pueden solicitar a Andami la traducción de una clave. Andami devolverá la traducción para el idioma elegido en el panel de configuración del idioma. Mediante este mecanismo es posible presentar al usuario una interfaz totalmente traducida al idioma seleccionado.
- *Servicio de registro (log):* Permite mantener un histórico de los sucesos de una sesión de gvSIG. Los sucesos pueden ser información, avisos, errores o información para depuración. En el registro de un mensaje en el log, se registra también automáticamente el plugin que ha generado el mensaje.
- *Persistencia de datos*: En el fichero *plugin-persistence.xml* los plugins pueden almacenar información. Por ejemplo un plugin de un servicio WMS puede almacenar la lista de servidores recientes. Al volver a arrancar gvSIG, posibilita a los plugins la recuperación de la información guardada. En el ejemplo el plugin podría recuprar y mostrar la lista de servidores usados durante la última sesión. Cada plugin sólo tiene acceso directo a la información que él mismo produce. El fichero de persistencia es adecuado para guardar pequeñas cantidades de

información de texto, e inadecuado para otros tipos de información como miniaturas de la cartografía, un listado amplio de nombres de municipios, etc.

- *Ficheros temporales*: Andami facilita la creación de ficheros temporales. Éstos son eliminados automáticamente al salir de gvSIG, siempre que gvSIG finalice correctamente. Si se proporciona un nombre y unos datos, Andami creará el fichero temporal con los datos proporcionados. También es posible solicitar la ruta del directorio de ficheros temporales y crear en él los ficheros necesarios. Para los dos casos, los ficheros serán eliminados automáticamente al salir de gvSIG.
- *Portapapeles*: Andami permite tanto obtener como depositar texto en el portapapeles. Para otros tipos de datos, se debe usar directamente métodos que Java ofrece para el acceso al portapapeles.
- *Tareas en segundo plano*: Andami facilita la creación y ejecución de tareas en segundo plano. Permitie que estas tareas sean cancelables. Si se pretende realizar un procesamiento prolongado, es conveniente lanzarlo en un proceso en segundo plano, y así no bloquear la interfaz gráfica. Es habitual en tareas de exportación de una capa a un fichero, ejecución de geoprocesos, etc. Se presentará un diálogo con el progreso de la tarea, y un botón que permita cancelar el proceso.
- *Accesso a recursos del plugin*: Andami hace posible el accea a recursos de los plugins como imágenes, ficheros XML, etc. mediante rutas relativas al directorio raíz del plugin, y así favorecer la portabilidad del código.

5.2.2.2 **Plugins y extensiones.**

En gvSIG instalado, existe un directorio llamado *gvSIG/extensiones*. Este directorio contiene a su vez un conjunto de directorios. Cada uno de éstos constituye un plugin. Los plugins son un conjunto de clases Java que incorporan nuevas funcionalidades a la aplicación. Así al inspeccionar el directorio observaremos la existencia de un plugin llamado *com.iver.core*, que contiene *libCorePlugin* (el skin de Andami) y un plugin llamado *com.iver.cit.gvsig*, que contiene el plugin gvSIG (plugin principal de la aplicación). Dependiendo de la instalación de

gvSIG que se haya realizado, será posible localizar otros plugins como *com.iver.cit.gvsig.cad*, *org.gvsig.scripting*, *com.iver.cit.gvsig.wms* que aportan funcionalidades de edición cartográfica, capacidades de scripting y un cliente de WMS respectivamente, y un largo etcétera.

Las nuevas funcionalidades que añaden los plugins, deben conectarse con el resto de la aplicación. Esta conexión la aportan las extensiones. Las extensiones son clases Java que implementan la interfaz IExtension, y hacen de puente entre la funcionalidad existente y la aportada por el plugin.

El contenido mínimo que un plugin debe tener se reduce a un fichero llamado config.xml. Este fichero contiene el nombre del plugin y una serie de datos relevantes como el directorio que contendrá las librerías que aporta el plugin, las dependencias con otros plugins, el nombre base de los ficheros de traducciones, las extensiones aportadas por el plugin, y una descripción de los componentes a añadir a la interfaz de usuario como menús, barras de herramientas, etc. Dependiendo de los elementos definidos en el fichero config.xml, el plugin puede incluir también uno o varios ficheros jar en el directorio de librerías, diversos ficheros de traducciones, imágenes o cualquier otro tipo de datos como son cartografía, datos de conexión a algún servicio, paletas de colores, etc.

Podemos ver la estructura del plugin WFS para hacernos una idea de una distribución típica de ficheros:

```
gvSIG/
extensiones/
    com.iver.cit.gvsig.wfs2/
    build.number
    config.xml
    text_en.properties
    text_fr.properties
    \.\.
    text.properites
    lib/
        com.iver.cit.gvsig.wfs2.jar
    images/
        backward.png
        down-arrow.png
        fastbackward.png
        fastforward.png
```

Las librerías se depositan normalmente en el directorio raíz del plu-

gin, o en un subdirectorio *lib*. Las imágenes se situan habitualmente en el subdirectorio *images*. Sin embargo es posible modificar esta estructura siempre que en el código use la ruta relativa correcta, para las imágenes, o en el caso de las librerías se defina correctamente el directorio en el fichero config.xml.

5.2.2.3 El fichero config.xml

La extensibilidad de Andami se basa en la seguiente filosofía: existen extensiones, que son clases que implementan la interfaz IExtension, la que incluye un método *execute()* entre otros. En el fichero de configuración del plugin *config.xml*, se definen elementos de interfaz de usuario que se van a incorporar a la aplicación como son herramientas, menús y controles de la barra de estado, y se le asocia a cada uno de estos elementos una extensión. De esta forma, al pinchar el usuario sobre una herramienta o una entrada de un menú, Andami ejecutará el método *execute* de la extensión asociada. De este modo, la extensión ejecutará la operación solicitada, apoyándose en otras clases o librerías del plugin posiblemente.

La extensión es el punto de acceso a las nuevas funcionalidades que aporta el plugin, y los elementos de interfaz de usuario, definidos en el config.xml, junto con la extensión asociada son el nexo entre la aplicación y el plugin. Un ejemplo de fichero config.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
  <depends plugin-name="com.iver.cit.gvsig" />
  <libraries library-dir="./lib"/>
  <resourceBundle name="text"/>
  <extensions>
    <extension class-name="com.iver.cit.gvsig.wfs.WFSClientExtension"
      description="Support to access WFS"
      active="true"
      priority="1">
    </extension>
    <extension
      class-name="com.iver.gvsig.centerviewpoint.CenterViewToPointExtension"
      description="Haces zoom a partir de un par de coordenadas"
      active="true">
        <menu text="Vista/Centrar_la_Vista_sobre_un_punto"
          tooltip="Centrar_la_Vista_sobre_un_punto"
          action-command="CENTERVIEWTOPOINT"
          icon="images/centerviewtopoint.png" />
        <tool-bar name="com.iver.cit.gvsig.Herramientas">
          <action-tool icon="images/centerviewtopoint.png"
            action-command="CENTERVIEWTOPOINT"
            tooltip="Centrar_la_Vista_sobre_un_punto"
```

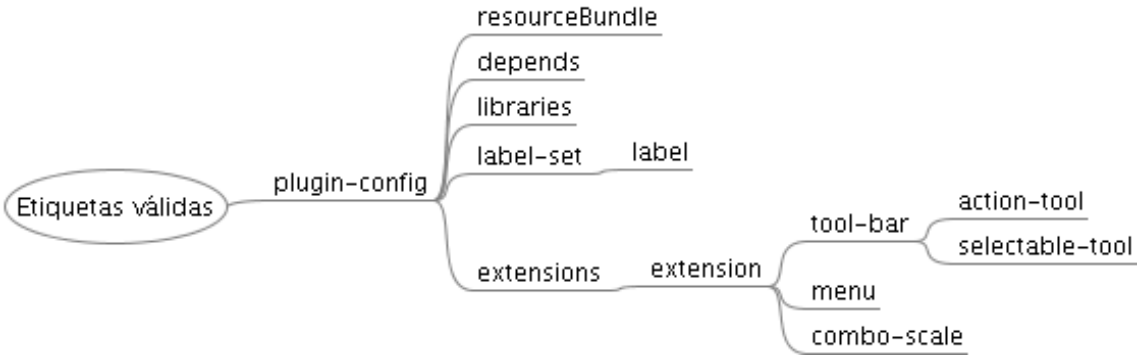


Figura 5.7: Jerarquía de etiquetas válidas de plugin-config.xml.

```
            enable-text="debería de estar activada" last="true"/>
        </tool-bar>
    </extension>
  </extensions>
</plugin-config>
```

El fichero utiliza una sintaxis XML. Esto implica una estructura marcada por etiquetas que deben abrirse y cerrarse en orden inverso al orden de apertura. Para más información sobre XML en el sitio web del W3 Consortium (*http://www.w3.org/*).

Se describen a continuación las etiquetas permitidas. Dentro de la distribución del código fuente de gvSIG, se encuentra un fichero *plugin-config.xsd*, que está ubicado dentro de *_fwAndami/schemas*, y se describen formalmente las etiquetas permitidas y tipo de valores que permiten. Este fichero está escrito en lenguaje XSD Schema, se trata de un lenguaje de descripción de sintaxis XML. El *plugin-config.xsd* debe constituir la referencia principal para conocer las etiquetas permitidas.

El fichero XML debe tener una cabecera como la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
```

El primer carácter del fichero es < y no tiene que haber ningún carácter en blanco antes de la cabecera, de no ser así el fichero se leerá incorrectamente. *version*: Indica la versión de XML que se va a utilizar, en el ejemplo la "1.0". *encoding*: Indica la codificación de caracteres del fichero. Debe coincidir con la codificación utilizada en el momento de gualdar el fichero (los editores de texto permiten normalmente especificarla en algún apartado). Codificaciones típicas son

"UTF-8", "ISO-8859-15" e "ISO-8859-1".

`<plugin-config>`

Esta etiquta marca el inicio y el fin del fichero de configuración. Es la raíz del árbol XML.

`<resourceBundle>`

Indica el nombre base de los ficheros de traducción. Por ejemplo si toma el valor <resourceBundle name="text"> indica que los ficheros de traducciones se llaman *text.properties*, *text_en.properties*, etc.

`<depends>`

Indica los plugins de los que depende el plugin, a nivel de librerías. Si se desea usar alguna clase o librería que estén en un plugin distinto, se debe declarar una dependencia de ese plugin. De este modo habrá una entrada <depends> por cada plugin del que dependa. Las dependencias determinan el orden de carga de plugins. Un plugin siempre se carga después de haber cargado sus dependencias. Habitualmente, los plugins dependen del plugin gvSIG (com.iver.cit.gvsig), que es el plugin que se carga primero.

`<libraries>`

Establece el directorio de las librerías del plugin. Habitualmente se utiliza <libraries library-dir="."> para el directorio raíz del plugin, o <libraries library-dir="lib"> para indicar el subdirectorio *lib*. Dos plugins diferentes pueden tener distintas versiones de la misma librería, y ambas funcionarán sin interferencias excepto en los siguientes casos:

- Si un plugin A declara una dependencia de un plugin B, y ambos poseen diferente versiones de la misma librería, el plugin B usará su propia versión sin ningún problema y el plugin A usará su propia versión si el plugin B se carga después de A, en caso contrario se usará la versión de B.
- Si existe un plugin A que tiene una librería, y ésta está en el CLASSPATH inicial de Andami (declarado al lanzar Andami), el

plugin A utilizará la versión presente en el CLASSPATH inicial.

`<label-set>`

Esta etiqueta indica el comienzo de un grupo de etiquetas, label, para la barra de estado. Cada grupo de etiquetas, label-set, asociado a una clase, y sólo visible cuando la ventana visible es instancia de dicha clase. Acepta el siguiente atributo:

- *class-name* nombre de la clase asociada a este label-set. Las etiquetas del label-set sólo serán visibles si la ventana activa es instancia de esta clase.

`<label>`

Instala una etiqueta, que se trata de texto no editable, en la barra de estado. La etiqueta pertenece siempre a un grupo de etiquetas, label-set, y sólo será visible cuando lo sea el label-set al que pertenece. Acepta los siguientes atributos:

- *id* Identificador de la etiqueta. Se usará para el acceso a la etiqueta y la escritura sobre ella.
- *size* Anchura de la etiqueta en píxeles.

`<extensions>`

Marca el inicio de la lista de extensiones.

`<extension>`

Sirve para declarar una extensión de Andami. La declaración incluye el nombre de la clase Java que implementa la extensión, y una serie de elementos de interfaz de usuario que se añadirán a la aplicación y estarán asociados con esta extensión. Acepta los siguientes atributos:

- *class-name* Nombre de la clase que implementa esta extensión. La clase debe estar ubicada en alguna librería incluida en el plugin.
- *description* Descripción de la funcionalidad que aportada por la extensión.

- *active* Establece si la extensión está activa o no. El hecho de no estar activa resulta como si no existiera.
- *priority* La prioridad de la extensión. Determina el orden de carga respecto a otras extensiones del mismo plugin. Una extensión con prioridad menor se carga antes que otra extensión con prioridad mayor. Las dependencias entre plugin son las que determinan el orden de carga de los plugins, por lo tanto si queremos es cambiar el orden de carga de dos extensiones que están en plugins distintos, se debe modificar las dependencias entre ellos, y no la prioridad. Se usan valores, por convención entre 1 y 99999 aunque se admiten rangos mayores. Para una extensión sin ninguna necesidad especial, es posible omitir el valor prioridad, o darle un valor de 2000.

```
<tool-bar>
```

Esta etiqueta crea una barra de herramientas. Dentro esta etiqueta se anidarán los botones que se pretenden incorporar a esta barra de herramientas. Distintas extensiones pueden añadir botones a la misma barra de herramientas, para ello deben llamarla igual. Acepta los siguientes atributos:

- *name* Nombre de la barra de herramientas. Si el nombre de la tool-bar de distintas extensiones es el mismo, se añadirán a la misma tool-bar. También se mostrará traducido al idioma correspondiente en la lista de barras de herramientas , por esto que name debe ser considerada como una clave de traducción.
- *position* Sirve para establecer la posición de la barra de herramientas respecto a otras. Un valor de position más bajo que la de otra tool-bar indica que se situará más a la izquierda que la de mayor valor. Si distintas extensiones definen positions diferentes para la misma tool-bar, se tendrá únicamente en cuenta la definida en la extensión con mayor prioridad, esto es, la de valor de priority más bajo.
- *is-visible* Determina la visibilidad inicial de la tool-bar. Es posible cambiarlo posteriormente por código, o por el usuario en los menús.

```
<action-tool>
```

Una action-tool se trata de una herramienta, por ejemplo un botón, que lanza una acción determinada. El mecanismo consiste en los siguientes pasos: el botón es pulsado por el usuario, el evento de pulsado lo recibe Andami, quien consulta la extensión asociada al botón y ejecuta el método *execute(String actionCommand)*, de la extensión, pasándole como parámetro el valor de *action-command* definido para el botón. Acepta los siguientes atributos:

- *action-command* Especifica el comando que se desea ejecutar por la extensión asociada. Puede omitirse si la extensión va a tener un solo botón o menú, asociado, o si todos los botones, menús, etc ejecutan la misma acción.
- *name* Nombre del botón. De utilidad si se desea posteriormente obtener el botón para cambiarle, por ejemplo, el icono.
- *position* Para establecer la posición del botón en de la barra de herramientas. Un valor position más bajo que la de otro botón situará más a la izquierda que el mayor. En el caso de existir diferentes botones con la misma posición o sin posición, se ordenarán de manera arbitraria. Un botón con posición irá a la izquierda de uno sin posición. De todas formas, no es recomendable la omisión de este atributo.
- *icon* Icono que aparecerá dentro del botón. La ruta de la imagen debe ser relativa al directorio raíz del plugin.
- *text* Texto opcional a mostrar junto al icono.
- *tooltip* Texto que se mostrará en el globo de texto mostrado al situar el ratón sobre el botón durante unos segundos.
- *enable-text* Texto informativo de las condiciones para que el botón esté activo. Si el botón está visible pero desactivado, el tooltip muestra este texto en lugar del normal, para informar al usuario de lo que debe hacer para poder utilizar esta herramienta.
- *last* Sirve para dejar un espacio extra en la parte derecha de este icono. Suele usarse en el último icono de una barra de herramientas, y así marcar más la separación respecto a la barra de her-

ramientas siguiente.

```
<selectable-tool>
```

Una selectable-tool es un botón que puede estar pulsado o no pulsado, y no ejecuta una acción como la action-tool, sino que indica un cambio de estado. Un ejemplo de selectable-tool es el Zoom+, Zoom-. Cuando una selectable-tool está pulsada, las demás están no pulsadas. Al pulsar en otra selectable-tool, la anterior pasa a no estar pulsada. La selectable-tool seleccionada va asociada a la ventana activa: al cambiar de ventana activa, se carga la selectable-tool seleccionada asociada a la nueva ventana activa. Al volver a la ventana anterior, pasa a seleccionarse la selectable-tool asociada a la ventana anterior. La etiqueta selectable-tool admite los mismos atributos que la etiqueta action-tool, con la misma semántica, a excepción de action-command que en este caso indica qué herramienta está seleccionada. Adicionalmente selectable-tool permite:

- *is-default* Determina si la herramienta estará seleccionada inicialmente.

```
<menu>
```

Esta etiqueta sirve para crear una entrada de menú. El menú funcionará como una action-tool, al pinchar en el menú se ejecutará el método *execute(String actionCommand)* de la extensión asociada, como parámetro se le pasa el action-command del menú. Admite los siguientes atributos:

- *action-command* Indica el comando que se desea ejecutar en la extensión asociada. Es posible omitirlo si la extensión va a tener un único botón o menú, etc. asociado, o bien si todos los botones, menús, etc. ejecutan la misma acción.
- *text* Determina la localización y el texto que se mostrará en la entrada del menú. La etiqueta text tiene la forma "Archivo/Abrir". Cada porción separada por la barra (/) indica un contenedor de menú, como Archivo, a excepción de la última porción que indica el texto que se mostrará en la propia entrada. En el ejemplo "Archivo/Abrir" creará una entrada Abrir en el menú Archivo. Otro ejemplo: "Archivo/Plantillas/Abrir Plantillas" creará una

entrada Abrir Plantillas dentro del submenú Plantillas, dentro del menú Archivo. Si los menús contenedores no existieran, se crearían de forma automática. El texto real mostrado en los menús y submenús se traduce por lo que cada porción de texto utilizado aquí constituye una clave de traducción.

- *position* Se utiliza para establecer la posición de la entrada de menú en el submenú en el que se ubica. Un valor position más bajo que el de otra entrada situará más arriba la entrada, es decir, más cerca del inicio del menú. En el caso de existir diferentes entradas con la misma posición, o sin posición, se ordenarán de forma arbitraria. Una entrada con posición estará más arriba que una entrada que no tenga posición. Es recomendable no omitir este atributo.

- *icon* Icono a mostrar en el botón. La ruta de la imagen es relativa al directorio raíz del plugin.

- *tooltip* Texto que se muestra en el globo de texto, tooltip, que se muestra al situar el ratón sobre el botón durante unos segundos.

- *enable-text* Texto informativo de las condiciones que activarían el botón. Si el botón está visible pero también desactivado, el tooltip muestra este textoen lugar del normal, para informar al usuario de la manera de activar esta herramienta.

- *key* La key es el carácter que forma la combinación de teclas que lanza este menú. Se forma utilizando el modificador del sistema operativo + key. Por ejemplo en Windows el modificador es la tecla ALT, y la key de la entrada de menú Añadir capa es O, por lo tanto pulsando ALT+O aparece el diálogo de añadir capa.

- *mnemonic* El mnemonic es la tecla que activa esta entrada de menú cuando éste esté desplegado y tenga el foco. Por ejemplo, si el menú Archivo está seleccionado, y pulsamos la tecla A, aparece el diálogo de abrir proyecto ya que A es el mnemotécnico de la entrada Abrir proyecto.

- *is*_separator Atributo que añade un separador en esta posición de la barra de menús. Debe añadirse en una entrada de menú aparte, puwato que no añade un separador bajo la entrada actual, sino

que transforma la entrada actual en un separador, por tanto no se mostrará texto ni icono, ni se podrá pulsar sobre ella.

```
<combo-scale>
```

Añade a la barra de estado un control de tipo combo box. Se trata de un control que permite elegir un valor de una lista de valores, o bien escribir el valor deseado. Aparece además una etiqueta de texto descriptiva a la izquierda. Acepta los siguientes atributos:

- *name* Nombre del control. Debería ser distinto al nombre de cualquier otro control.
- *label* El texto que se muestra a la izquierda del combo box.
- *elements* Lista de valores numéricos separados por punto y coma. Cada valor representa una entrada del combo box.
- *value* Valor inicialmente seleccionado. En caso de omitirse aparece el primer elemento seleccionado.
- *action-command* Especifica el comando que se pretende que ejecute la extensión asociada. Si la extensión va a tener un solo botón, o menú, etc. asociado se puede omitir, del mismo modo que si todos los botones, menús, etc. ejecutan la misma acción.

5.2.2.4 Extensiones

Las extensiones son clases puente entre la funcionalidad de gvSIG y las funcionalidades que aporta el plugin. La extensión recibe notificación de las acciones realizadas, y ejecuta el código apropiado en respuesta a estas acciones, posiblemente usando otras clases del plugin. El siguiente diagrama (ver imagen 5.10) muestra un esquema de todas las clases implicadas en la creación de extensiones.

Todas las extensiones en gvSIG implementan la interfaz IExtension, que incluye los siguientes métodos:

- *public void initialize()* Una vez leidas y ordenadas todas las extensiones presentes en los plugins, este método es invocado para cada una de ellas, secuencialmente atendiendo al orden definido en la prioridad de las extensiones.

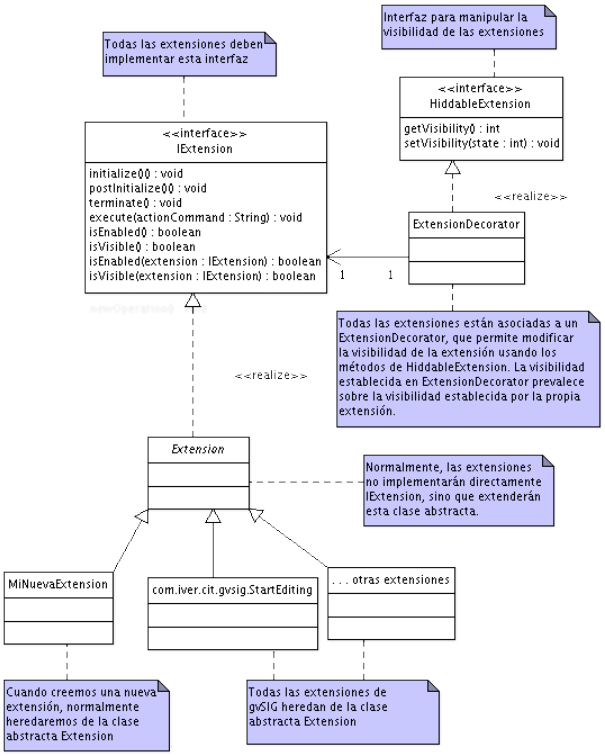


Figura 5.8: Diagrama de las clases implicadas en la creación de extensiones.

- *public void postInitialize()* Una vez inicializadas todas las extensiones, se llama a este método para cada extensión, siguiendo el mismo orden de antes.

- *public void terminate()* A la salida de la aplicación, se invoca a este método para cada extensión, siguiendo un orden secuencial inverso al seguido en la inicialización. La última extensión inicializada es la primera que se finalizará, y la primera que se inicializó será la última en ser finalizada.

- *public void execute(String actionCommand)* En el fichero config.xml se definen elementos de la interfaz de usuario como botones, menús, etc. asociados a una extensión. Al pulsar en uno de estos elementos de la interfaz, se busca la extensión asociada y se invoca a este método. Los botones, menús, etc pueden tener asociada una cadena de texto que actúa como identificador de comando llmada action command. Al llamar al método execute, se pasa como parámetro el actionCommand asociado al botón pulsado. De este modo, si existen varios botones, menús, etc. asociados a esta extensión, es posible discriminar mediante una comprobación

del actionCommand recibido. Normalmente, será en este método donde se encuentran las acciones principales de la extensión (abrir un asistente, comenzar algún procesamiento, etc).

- *public boolean isEnabled()* Los elementos de interfaz de usuario tales como botones, menús, controles de la barra de estado, etc. asociados a esta extensión estarán activados, o desactivados en función del valor devuelto por este método. Si un botón está desactivado, aparece en gris claro y no se puede pulsar sobre él.

- *public boolean isVisible()* Los elementos de interfaz de usuario tales como botones, menús, controles de la barra de estado, etc. asociados a esta extensión estarán visibles o invisibles en función del valor devuelto por este método. Si un botón está invisible, no se mostrará independientemente del valor que devuelva el método isEnabled(). Tanto este método, isVisible(), como el anterior, isEnabled(), determinan si la extensión y sus elementos de interfaz de usuario asociados serán visibles y estarán activos. Estas propiedades se comprueban para todas las extensiones cada vez que cambia la ventana activa y también justo después de procesar un evento de interfaz de usuario, es decir, después de invocar al método execute de una extensión.

- *public boolean isEnabled(IExtension extension)* Decide si la extensión que se pasa como parámetro estará activada o no, ignorando el valor devuelto por el método isEnabled de la propia extensión.

- *public boolean isVisible(IExtension extension)* Decide si la extensión que se pasa como parámetro estará visible o no, ignorando el valor devuelto por el método isVisible de la propia extensión. Tnto este método como el anterior sólo se usan con el mecanismo ExclusiveUIExtension. Esto permite a una extensión tomar el control de la interfaz y poder activar o desactivar otras extensiones, es decir mostrarlas u ocultarlas. Permite crear extensiones que personalicen gvSIG por ejemplo, una extensión que convierta gvSIG en un simple visor de cartografía.

Salvo que exista una necesidad especial, normalmente no se implementará directamente la interfaz IExtension, sino que se extenderá la clase abstracta Extension. Ésta a su vez ya implementa IExtension.

Esto ofrece cierta tranquilidad de futuro en el funcionamiento de la extensión, puesto que si se añadiera algún método a la interfaz IExtension, este método se implementaría en la clase Extension de modo que la extensión continuaría funcionando sin cambios a menos que necesitara un comportamiento para el nuevo método diferente al implementado en Extension.

El mecanismo ExclusiveUIExtension permite poder hacer algo parecido a lo que se consigue mediante ExtensionDecorator, solo que en este caso existe una única extensión que decidirá el estado de las demás extensiones. ExtensionDecorator permite que cualquier extensión modifique el estado de las demás. ExclusiveUIExtension permite definir una única extensión que podrá mostrar, ocultar, activar y desactivar el resto de extensiones. Ambos mecanismos pueden ser usados al mismo tiempo. ExclusiveUIExtension respetará lo definido por ExtensionDecorator, de modo que si ExtensionDecorator define una visibilidad como ALWAYS_VISIBLE, siempre visible, la extensión estará visible pese a lo definido en ExclusiveUIExtension. Esto significa que ExclusiveUIExtension actuará sólo en caso de que el ExtensionDecorator esté inactivo, es decir, modo ExtensionDecorator.INACTIVE. En cualquier caso, ExtensionDecorator sólo actúa sobre la visibilidad de la extensión, mientras que ExclusiveUIExtension puede definir además si la extensión está activa, es decir enabled, o no.

Para utilizar ExclusiveUIExtension se necesita:

- Una extensión que implemente los métodos isEnabled(IExtension extension) e isVisible(IExtension extension). La extensión actuará como ExclusiveUIExtension, decidiendo el estado visible, invisible, activa o desactivada, del resto de extensiones.
- Registrarse como ExclusiveUIExtension en Andami, dentro del método initialize de la extensión, empleanto el método *PluginServices.setExclusiveUIExtension()*.
- Otra modo equivalente al uso del método setExclusiveUIExtension es sumistrar un parámetro de arranque a gvSIG. El parámetro es: *ExclusiveUIExtension=NombreDeExtension*, donde NombreDeExtension es el nombre de la extensión mencionada en el punto anterior.

- En Linux, esto se puede conseguir editando el lanzador gvSIG.sh añadiéndole este parámetro antes del símbolo "$@", o también pasándole el parámetro al lanzador.
- En Windows es posible modificar el fichero gvSIG.ini, en la línea que empieza por command, incorporando el parámetro justo antes del símbolo ARGS. También se puede pasar el parámetro al comando gvSIG.exe en el momento de su ejecución.

5.2.2.5 Gestión de ventanas de Andami

Andami permite la creación y gestión de ventanas, para ello ofrece una interfaz de ventana que se abstrae de los tipos de ventanas de Java Swing. De este modo, para crear una ventana, simplemente se crea un panel que implemente la interfaz IWindow, en lugar de pensar en ventanas de Swing, y se le solicita a Andami que cree una ventana con las propiedades deseadas. En función de las propiedades solicitadas y el skin instalado, creará un tipo de ventana distinto como JInternalFrame, JDialog, etc. pero esto será indiferente. Para no confundir la ventana real, que crea Andami, con el contenido, que se ha creado y que implementa la interfaz IWindow, a partir de ahora utilizaremos los términos frame, marco o ventana real para hacer referencia a la ventana que crea Andami, y ventana para hacer referencia al contenido, que implementa IWindow.

Podemos hacer una primera distinción entre las ventanas de Andami: ventana principal, y resto de ventanas. La ventana principal, conocida como MainFrame, contiene las barras de herramientas, barra de estado y ,en algunas plataformas, la barra de menús. El gestor de ventanas no controla esta ventana, sino que se accede a ella utilizando un método de la clase PluginServices específico. Las demás ventanas sí están bajo el control del gestor de ventanas, MDIManager, al que dirigirse para poder crearlas, cerrarlas o cambiar sus propiedades.

La interfaz *MainFrame* representa la *ventana principal* de gvSIG. Es la ventana a la que se añaden menús, barras de herramientas, etc. La clase que la implementa, MDIFrame, hereda de JFrame, e incluye el JDesktopPane utilizado para contener las ventanas internas como

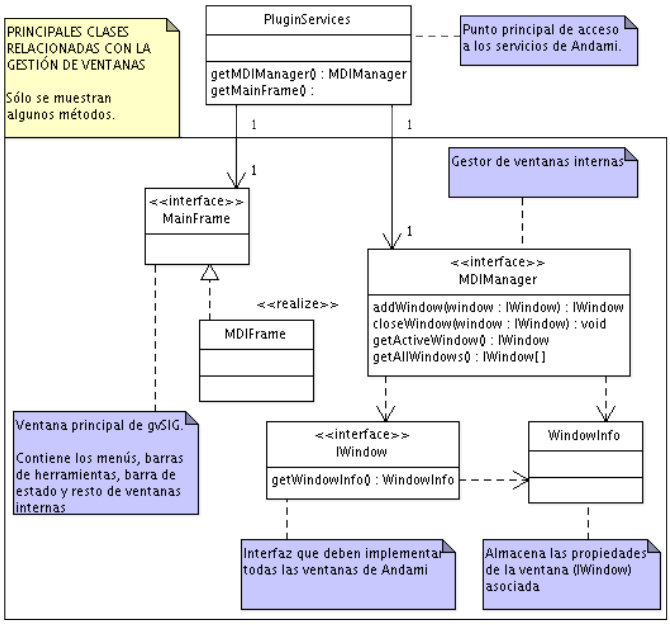


Figura 5.9: Diagrama de las principales clases implicadas en la gestión de ventanas.

vistas, mapas, etc. Normalmente, Andami crea automáticamente la interfaz de usuario a partir de los config.xml de los plugins. No obstante, MainFrame dispone de métodos para la modificación por código de algunas partes de la interfaz.

- Mediante los métodos *addMenu()*, *changeMenuName()* y *removeMenu()* es posible alterar los menús de gvSIG.
- El método *getJMenuBar()* permite acceder a la barra de menús, lo que permite alterar los menús.
- El método *getStatusBar()* permite el acceso a la barra de estado.
- *setTitle()* permite cambiar el título de la ventana.
- *setStatusBarLabels(Class clase, Label[] labels)* y *removeStatusBarLabels(Class clase)* añaden o eliminan etiquetas de la barra de estado. Estas etiquetas etiquetas se asocian a una clase, objeto tipo Class, y serán visibles si la ventana activa es un objeto de esa clase.
- *getComponentByName(String name)* obtiene un componente a partir de su nombre. De este modo es posible la obtención de barras

de herramientas, controles de la barra de herramientas o de la barra de estado, y menús.

- *addStatusBarControl()* y *removeStatusBarControl()* permiten incorporar controles personalizados a la barra de estado que serán visibles y activos cuando la extensión asociada lo esté.

La barra de estado, *NewStatusBar* es la banda situada en la parte inferior de la ventana principal. Contiene etiquetas y algunos controles. Se puede cambiar el texto de las etiquetas existentes, así como incorporar etiquetas o controles.

- public void setMessage(String id, String nuevoTexto) Permite cambiar el texto de la etiqueta id por nuevoTexto. id debe corresponder con el atributo id de una etiqueta definida en el config.xml o en el caso de un control, con su nombre.
- *setInfoText(String text)* Añade un mensaje de información en la barra de estado en la parte izquierda, siempre que no se esté mostrando un mensaje temporal. En ese caso, el mensaje establecido ahora se mostrará al llamar al método *restaurarTexto()*.
- *setWarningText(String text)* Añade un mensaje de aviso en la barra de estado, siempre que no se esté mostrando un mensaje temporal. En ese caso el mensaje establecido se mostrará al llamar al método *restaurarTexto()*.
- *setErrorText(String text)* Muestra un mensaje de error en la barra de estado, siempre que no se esté mostrando un mensaje temporal. En ese caso, el mensaje establecido ahora se mostrará al llamar al método *restaurarTexto()*.
- *setInfoTextTemporal(String text)*, setWarningTextTemporal(String text), setErrorTextTemporal(String text) y restaurarTexto() Muestran mensajes de información, aviso u error respectivamente, pero sólo temporalmente, pues si a continuación se utiliza el método *restaurarTexto()* se vuelve a mostrar el mensaje anterior.
- *setProgress(int progress)* Si progress está entre 0 y 99 inclusive muestra una barra de progreso situada en la izquierda y avanza

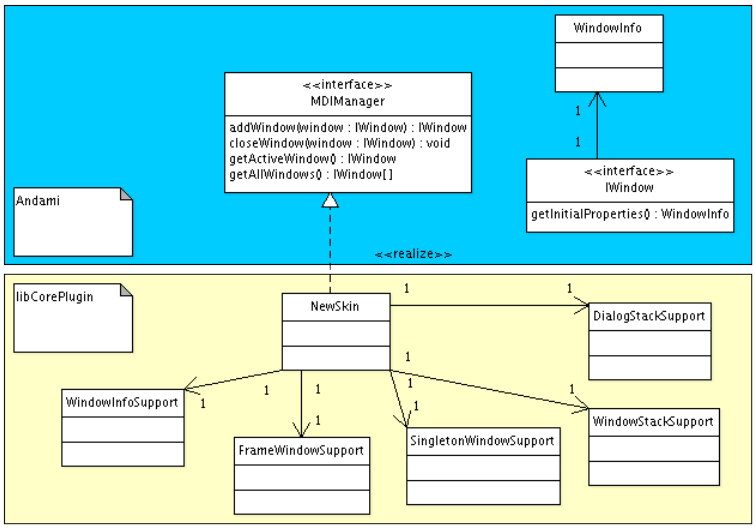


Figura 5.10: Clases principales de CorePlugin y clases relacionadas de Andami.

la posición a ese valor. Si progress supera o iguala el valor 100 se oculta la barra de progreso.

La gestión de plugins se incorpora directamente en Andami, sin embargo la gestión de ventanas se delega en los *skins* de Andami. Andami ofrece interfaces y objetos que permiten la gestión de ventanas: IWindow, MDIManager, IWindowListener, WindowInfo, SingletonWindow,etc. Esas interfaces deben ser implementadas por el skin, el que realizará la gestión de ventanas. De este modo, gestionar las ventanas de otro modo, implementando un skin distinto. En la actualidad sólo existe un Skin implementado, *CorePlugin*. De manera formal es un plugin más de Andami, aunque con la particularidad de que Andami no arrancará si CorePlugin, u otro skin, no está instalado. Los skins tienen que incluir una extensión especial que implemente IExtension y MDIManager, que se define en el config.xml mediante la etiqueta *skin-extension*.

Cualquier ventana que deseemos crear mediante Andami debe implementar la interfaz *IWindow*. IWindow no será ventana de Swing, sino el contenido de la ventana, que normalmente se trata de una clase que hereda de JPanel. Andami crea la ventana propiamente dicha (frame) y no se dispone de acceso directo a ella. La interfaz IWindow posee un único método que especifica las propiedades iniciales de la ven-

tana creada como son: posición, tamaño, tipo, título, etc. Se trata del método *public WindowInfo getWindowInfo()*. Andami lo invoca en la creación de la ventana real para determinar sus propiedades y el tipo de ventana a crear. Devuelve un objeto tipo WindowInfo, el que describe las propiedades de las ventanas de Andami. Este método sólo devuelve las propiedades iniciales. Por eso para la obtención o modificación de las propiedades de la ventana es preciso dirigirse al gestor de ventanas. La mayoría de las ventanas creadas no implementarán directamente IWindow, sino SingletonWindow, una interfaz más específica que asocia un modelo a cada ventana. Este modelo es usado para dotar de identidad a las ventanas, y así al intentar crear una ventana con un modelo que ya esté asociado a otra ventana, obtendremos una referencia a esta última ventana.

El objeto *WindowInfo* almacena las propiedades de una ventana de Andami, existe un objeto WindowInfo asociado a cada ventana IWindow creada por Andami. Se usa para consultar propiedades actuales de la ventana y para modificarlas. La modificación de una propiedad de WindowInfo se refleja de forma inmediata en la ventana asociada. Existen ventanas de tres tipos en función de su comportamiento frente a otras: normales, modales y modeless.

- La ventana *modal* es aquella que siempre está sobre las otras y además no permite que el foco pase a ninguna de las otras. Al crear una ventana modal, no es posible interactuar con los menús, herramientas, etc., de la ventana principal hasta su cierre. Las ventanas modales, además, suspenden el hilo de ejecución principal de la aplicación, es decir, el código de las extensiones no continuará ejecutándose hasta el cierre de la ventana modal.
- La ventana *modeless* es aquella que siempre está encima pero sí permite que el foco pase a otras ventanas, e interactuar con los menús, herramientas de la ventana principal y con otras ventanas.
- Las ventanas *normales* pueden estar encima o debajo de las otras normales. No imponen ninguna restricción especial en cuanto al foco o a la superposición a otras ventanas.

Las ventanas modeless sólo lo serán respecto a las normales; es decir, una modeless siempre estará encima de las normales existentes, pero podrá estar encima o debajo de otra ventana modeless.

Las ventanas modales sólo lo serán respecto a las normales o modeless; es decir, estarán encima de las normales o modeless existentes, pero podrán estar encima o debajo de otras modales.

Para la creación de una ventana modal, el método *getWindowInfo()* de la ventana debe contener el siguiente código:

```
WindowInfo windowInfo = new WindowInfo(WindowInfo.MODALDIALOG);
return windowInfo;
```

Para la creación de una ventana modeless:

```
WindowInfo windowInfo = new WindowInfo(WindowInfo.MODELESSDIALOG);
return windowInfo;
```

Para la creación de una ventana normal:

```
WindowInfo windowInfo = new WindowInfo();
return windowInfo;
```

Es posible especificar otras propiedades en el constructor, que son añadidas utilizando el operador de suma binaria |:

- *WindowInfo.RESIZABLE* Permitirá al usuario redimensionar la ventana.De no incluirse, la ventana no será redimensionable y tendrá tamaño fijo.
- *WindowInfo.MAXIMIZABLE* La ventana podrá ser maximizable. Una ventana maximizable tiene dos modos: maximizado, en el que se empotra en la ventana principal de Andami, que ocupa todo su espacio, y modo no maximizado, en el que la ventana posee su tamaño normal. El objeto WindowInfo tiene también unos métodos para el cambio de este estado a nivel de código. De no especificarse lo contrario la ventana no será MAXIMIZABLE, es decir, siempre estará en el estado no maximizado.
- *WindowInfo.ICONIFIABLE* Permite al usuario minimizar la ventana. Al minimizar la ventana, desaparece y en su lugar aparece un botón en la ventana principal en la parte inferior izquierda. Al pulsar sobre este botón, la ventana vuelve a su estado normal.

- *WindowInfo.PALETTE* Ciertas ventanas están diseñadas para poder ser empotradas dentro de otras. Éstas tienen dos modos: empotrado y paleta flotante. En modo paleta, el contenido es mostrado en una ventana independiente, de tipo MODELESSDIALOG. En modo empotrado, el contenido aparece integrado en algún área de otra ventana. Por ejemplo, el localizador gráfico es una herramienta capaz de mostrarse en una ventana independiente o empotrado en la vista en la parte inferior izquierda. Las ventanas de tipo paleta deben implementar la interfaz *IWindowTransform*.
- Además, existen los dos atributos que ya hemos explicado anteriormente: *WindowInfo.MODALDIALOG* y *WindowInfo.MODELESSDIALOG*.

Para la creación de una ventana modeless redimensionable y maximizable:

```
WindowInfo windowInfo = new WindowInfo(WindowInfo.MODELESSDIALOG | WindowInfo.RESIZABLE |
    WindowInfo.MAXIMIZABLE);
return windowInfo;
```

Simultáneamente las propiedades MODELESSDIALOG y MODALDIALOG no es posible utilizarlas. Se obtendrá un error de Andami.

Hasta aquí se han visto las propiedades aceptadas por el constructor de WindowInfo, que determinan el comportamiento de la ventana. Existe una serie de métodos, además, que permiten consultar propiedades de la ventana y modificarlas. Permiten obtener la ubicación de la ventana en la pantalla, modificarla, consultar sus dimensiones, modificarlas, etc. Se trata de funciones muy intuitivas con las que el programador se familiarizará rápidamente.

La interfaz *MDIManager* permite interactuar con las ventanas. Es el punto al que dirigirse para la creación de ventanas, cierre, redimensionado, obtención de la ventana activa, obtención de la lista de ventanas, etc. En la creación de ventanas, se encarga de crear las ventanas con unas propiedades homogéneas, teniendo en cuenta las propiedades solicitas, y registrarlas en el menú Ventana. También escucha los cambios que se produzcan en los objetos WindowInfo para reflejarlos en la ventana real asociada. La gestión de ventanas de Andami es bastante particular ya que el programador únicamente crea un panel con

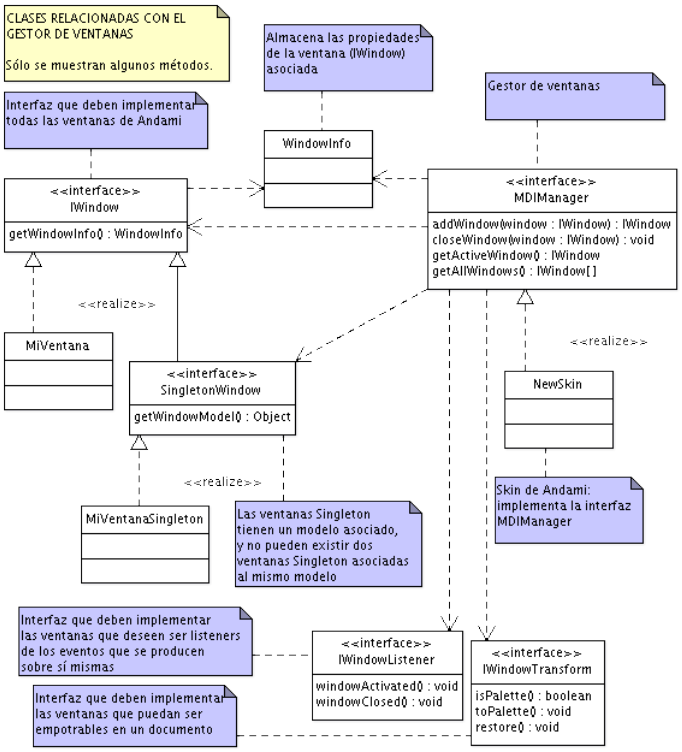


Figura 5.11: Diagrama que muestra las principales clases relacionadas con el gestor de ventanas.

el contenido de la ventana, y es MDIManager quien crea la ventana real. Por este motivo, para cambiar las propiedades de una ventana hay que dirigirse a MDIManager, quien conoce la implementación real de la ventana frame.

MDIManager tiene un amplio rango de métodos de interacción con las ventanas. La mayor parte de ellos requieren una referencia al objeto IWindow que se pretende alterar. Otros devuelven una referencia a tal objeto. Entre los métodos más destacados se encuentran:

- Métodos de creación de ventanas:
  - *public IWindow addWindow(IWindow panel)* Crea un frame cuyo contenido será panel. Las propiedades iniciales de la ventana como tamaño, tipo, etc. estarán determinadas por el método *getWindowInfo()* de panel. Si la ventana es de tipo MODAL, se centrará, ignorando la posición de *getWindowInfo*. Si es de tipo SingletonWindow y ya existe una ventana con el mismo modelo, no se creará sino que la ya existente se traerá a primer plano con el foco. En este caso se devolverá una referencia a

la ventana ya existente.

- *public IWindow addCentredWindow(IWindow panel)* Idéntico a addWindow, salvo que crea todas las ventanas y las sitúa en posición centrada, ignorando la posición especificada por *getWindowInfo()* de panel.

- Métodos de cierre de ventanas:
  - *public void closeAllWindows()* Cierra todas las ventanas que están abiertas.
  - *public void closeWindow(IWindow window)* Cierra la ventana *window*.
  - *public boolean closeSingletonWindow(Object model)* Cierra la ventana Singleton con modelo el objeto *model*.
  - *public boolean closeSingletonWindow(Class viewClass, Object model)* Cierra la ventana Singleton con la clase y modelo pasados como parámetros.

- Métodos para la obtención de ventanas existentes:
  - *public IWindow[] getAllWindows()* Obtiene un vector con todas las ventanas abiertas actualmente, incluyendo las minimizadas y maximizadas. No está incluída la ventana principal, pues no está bajo el control de MDIManager.
  - *public IWindow[] getOrderedWindows()* Obtiene un vector con todas las ventanas que están abiertas actualmente, ordenado según la profundidad de las ventanas. La primera ventana del vector será la situada primer plano, delante de las demás, y la última ventana del vector la que esté más al fondo.
  - *public getActiveWindow()* Devuelve la ventana activa, excluyendo las ventanas modales y las ventanas de tipo PALETTE. Si la ventana activaes modal o tipo PALETE, devuelve la última ventana activa que no fuese ni modal ni tipo PALETTE. Esto sucede así porque se consideran las ventanas modales y tipo PALETTE ventanas auxiliares que normalmente contienen alguna herramienta para la manipulación del contenido de otra ventana o presentan alguna información puntual).
  - *public getFocusWindow()* Devuelve la ventana que tiene el foco, excluyendo las ventanas modales. Si la ventana que posee el foco es modal, devuelve la última no-modal que tuvo el foco.

- Otros métodos destacables:
  - *public WindowInfo getWindowInfo(IWindow window)* Obtiene el objeto WindowInfo asociado a la ventana *window* que contiene información actualizada sobre la posición, tamaño, título, etc. Proporciona información y además puede utilizarse para realizar cambios en las propiedades de la ventana como tamaño, posición, etc.
  - *public void changeWindowInfo(IWindow window, WindowInfo wi)* Actualiza propiedades de la ventana *window* con los valores especificados en el objeto *wi*.
  - *public void refresh(IWindow win)* Normalmente, las ventanas se redibujan automáticamente si corresponde. No obstante, si queremos forzar un redibujado, usamos este método.

Hay un tipo de especial de ventanas de Andami: las *Singleton*. Estas ventanas que tienen asociado un modelo. Éste puede ser cualquier objeto Java, y sirve para dotar de identidad a la ventana. Existirá una única ventana asociada a un modelo. Será posible referenciarla a través del modelo. Si se intenta añadir una ventana Singleton cuyo modelo ya está asociado a otra ventana existente, en lugar de crearse una nueva ventana, se mandará la existente a primer plano, obteniendo una referencia a dicha ventana. La interfaz Singleton cuenta simplemente de un método:

- *public Object getWindowModel()* Devuelve el modelo de la ventana Singleton.

La interfaz *SingletonWindow* extiende la interfaz IWindow. así cualquier ventana que implemente SingletonWindow implementará también IWindow.

La interfaz *IWindowListener* permite a las ventanas que la implementen registrarse como oyentes de los eventos sucedidos sobre sí mismas. Dos métodos importantes:

- *public void windowActivated()* Se ejecutará cada vez que se active la ventana.
- *public void windowClosed()* Se ejecutará cuando la ventana se esté cerrando.

Para poder recibir estos eventos, basta con que las ventanas implementen esta interfaz, y de este modo Andami lo detectará y les enviará los eventos sin tener que seguir ningún paso adicional.

Las ventanas tipo PALETTE se tratan de ventanas que pueden empotrarse dentro de otras o estar en modo flotante o paleta. Estas ventanas deben implementar la interfaz *IWindowTransform*, para pasar de modo paleta a modo empotrado. Esta interfaz tiene tres métodos, que la ventana debe implementar:

- *public void toPalette()* Solicita a la ventana que se transforme en modo flotante.
- *public void restore()* Solicita a la ventana que se transforme en modo empotrado.
- *isPalette()* Devuelve true si la ventana está en modo paleta flotante, false si está en modo empotrado.

5.2.2.6 Servicios a los plugins

Son destacables los siguientes servicios que ofrece Andami a los plugins:

- La clase PluginServices
- Traducciones
- Registro (log)
- Persistencia de datos (plugin-persistence.xml)
- Tareas en segundo plano
- Ficheros temporales
- Accesso a recursos del plugin
- Acceso al portapapeles

Andami proporciona variedad de servicios a los plugins, prácticamente todos a través de la clase PluginServices. Éstos cubren tareas realizadas frecuentemente en cualquier parte de la aplicación, y por ello conviene ofrecer una manera sencilla de realizarlas. Cada plugin tiene un objeto tipo PluginServices asociado, al que acceder con la llamada:

```
PluginServices.getPluginSerivices(pluginName)
```

o también:

```
PluginServices.getPluginServices(object)
```

donde *object* debe ser algún objeto perteneciente al plugin. PluginServices permite el acceso a servicios específicos del plugin, como la obtención de recursos como imágenes, el nombre del plugin, la persistencia de datos, etc. Hay otros servicios no asociados específicamente a cada plugin accesibles mediante métodos estáticos de esta misma clase.

Los plugins tienen unos ficheros de *traducciones* leídos al inicio de la aplicación. Éstos contienen claves de traducción, y la traducción a cada clave, para cada idioma. Así pueden solicitar la traducción de una clave y Andami devolverá la traducción para el idioma elegido. Mediante este mecanismo es posible presentar al usuario una interfaz totalmente traducida a su idioma. Esttos ficheros de traducción deben situarse en el directorio raíz del plugin, y deben llamarse según lo especificado en la etiqueta resourceBundle del *config.xml*. Los sufijos del idioma de cada nombre de fichero se corresponden con el código ISO 639-1, *en* para inglés, *fr* para francés, etc de dos caracteres. El fichero que no tiene sufijo se corresponde con el idioma Castellano. Las traducciones del plugin se cargan automáticamente en la carga de plugins. Para acceder a ellas desde el plugin, se utiliza el código siguiente:

```
PluginServices ps = PluginServices.getPluginServices(this);
ps.getText("clave_de_traduccion");
```

o bien, a través de un método estático:

```
PluginServices.getText(this,"clave_de_traduccion");
```

Todos los plugins tienen un espacio de claves de traducción único. Si un plugin A tiene una traducción en el fichero text.properties así:

```
 error_conexion="Error de conexión"
```

y un plugin B tiene una traducción como:

```
 error_conexion="Error conectando a la base de datos"
```

entonces habrá un conflicto de claves. Si el plugin A se cargase antes, por tanto sus traducciones también, al intentar el plugin B acceder a la traducción:

```
PluginServices ps = PluginServices.getPluginServices(this);
ps.getText(error_conexion);
```

obtendría una traducción errónea: "Error de conexión \verb". Por ello utilizar claves que describan con precisión el mensaje para evitar este de error es muy importante. Si el plugin no va a ser integrado en la distribución principal de gvSIG, podemos usar un prefijo de clave para curarnos en salud:

```
org.gvsig.MipluginCasero.error_conexion="Error conectando a la base de datos"
```

El servicio de *registro (log)* permite guardar un histórico de los sucesos de una sesión completa de gvSIG. Los sucesos pueden ser de distinta categoría como son Información, Avisos, Errores o Información para Depuración. Cuando se registra algún mensaje en el log, se registra también automáticamente el plugin que lo genera. Registrar los errores en el log, a fin de poder depurarlos posteriormente es importante. Frecuentemente los usuarios encuentren errores que no se logran reproducir en otra máquina, y la única herramienta con la que corregirlos es mediante el log. El fichero de log se encuentra en el directorio gvSIG en el directorio del usuario. En Windows, típicamente en:

```
C:\Archivos de programa\Documents and Settings\$USER$\gvSIG\gvSIG.log
```

y en Linux, en:

```
/home/$USER/gvSIG/gvSIG.log
```

El log se utiliza desde un plugin así:

```
Logger logger = PluginServices.getLogger();
logger.info("Mensaje a mostrar en el log");
```

Existen 4 niveles de severidad:

- *depuración (DEBUG)*, indica información sólo interesante para la depurarción del programa. En esta severidad se pueden mostrar, por ejemplo, los típicos mensajes que verifican que el programa entra dentro de ciertas partes del código como un if.
- *información (INFO)*, indica un mensaje meramente informativo.
- *aviso (WARNING)*, indica un aviso de algún hecho especial. Deben mostrarse en esta severidad hechos inusuales o inesperados que pueden ser indicios de error.
- *error (ERROR)*, indica que se ha producido un error en la aplicación. Se debe mostrar información relevante para permitir la detección el origen del mismo y permitir subsanarlo, ya sea al

usuario o al programador.

Se accede a los cuatro niveles mediante los métodos: *debug(), info(), warning() y error()*.

Al producirse una excepción es posible incluir en el log la salida del método printStackTrace. Se hace mediante el siguiente código:

```
catch (Exception excepcion) {
  logger.error("Mensaje de error", excepcion);
}
```

Para la *persistencia de datos* se dispone de un fichero *plugin-persistence.xml* en el que pueden almacenar información los plugins. Por ejemplo un plugin de nomenclátor podría almacenar las últimas búsquedas realizadas. Los plugins pueden recuperar, al volver a arrancar gvSIG, la información guardada y presentarla. En el ejemplo sería posible mostrar el listado con las últimas búsquedas realizadas. Cada plugin únicamente tiene acceso directo a la información producida por él mismo. Se trata de un fichero adecuado para guardar únicamente pequeñas cantidades de información en texto. El fichero es leído de forma automática durante el inicio de Andami, es guardado automáticamente al cierre de la aplicación. Está en formato XML, formato que almacena la información con una estructura de árbol. El acceso para la lectura o escritura de información en esta estructura se realiza haciendo uso del objeto *XMLEntity*, el cual representa una subrama del árbol. Una subrama de un fichero XML puede contener desde un nodo único hasta el árbol completo. Para la obtención de la subrama asociada a un plugin se utiliza:

```
XMLEntity entity = PluginServices.getPluginServices(this).getPersistentXML();
```

Suponiendo que previamente se ha almacenado una propiedad llamada "last.server", se puede recuperar usando:

```
try {
  XMLEntity entity = PluginServices.getPluginServices(this).getPersistentXML();
  String lastServer = entity.getStringProperty("last.server");
}
catch (NotExistInXMLEntity exception) {
  // mostrar el error y tomar alguna acción en respuesta
}
```

El posible ver el objeto *XMLEntity* como una especie de tabla Hash que asocia claves con valores. En el anterior ejemplo , "last.server.[es] una clave. Para el almacenamiento de una nueva clave se usa:

```
XMLEntity entity = PluginServices.getPluginServices(this).getPersistentXML();
entity.putProperty("last.server", "http://www.gvsig.org");
```

Se pueden almacenar diversos tipos de objetos en el XMLEntity:

```
XMLEntity entity = PluginServices.getPluginServices(this).getPersistentXML();
entity.putProperty("last.server", "http://www.gvsig.org");
entity.putProperty("maxConnections", 5);
entity.putProperty("timeout", 3.258);
String[] servers = {"http://www.gvsig.org","www.gvsig.gva.es","www.cit.gva.es"}
entity.putProperty("preferred.servers", servers);
int[] allowedValues = {1,2,5,10};
entity.putProperty("allowed.values", allowedValues);
```

Y para la recuperación de estos objetos:

```
try {
  XMLEntity entity = PluginServices.getPluginServices(this).getPersistentXML();
  String lastServer = entity.getStringProperty("last.server");
  int maxConnections = entity.getProperty("maxConnections");
  float timeout = entity.getFloatProperty("timeout");
  String[] servers = entity.getStringArrayProperty("preferred.servers");
  int[] allowedValues = entity.getIntArrayProperty("allowed.values");
}
catch (NotExistInXMLEntity exception) {
  // mostrar el error y tomar alguna acción en respuesta
}
```

Además, es posible anidar los objetos XMLEntity:

```
XMLEntity entity = PluginServices.getPluginServices(this).getPersistentXML();
XMLEntity hijo = new XMLEntity();
hijo.setName("ColorData");
hijo.putProperty("red", "#ff0000");
hijo.putProperty("green", "#00ff00");
hijo.putProperty("blue", "#0000ff");
hijo.putProperty("black", "#000000");
hijo.putProperty("white", "#ffffff");
entity.addChild(hijo);
```

Y para la recuperación del XMLEntity anidado:

```
XMLEntity child=null, tmpchild;
for (int i=entity.getChildrenCount()-1; i>=0; i--) {
  tmpchild = entity.getChild(i);
  if (child.getName().equals("ColorData")) {
    child = tmpchild;
    break;
  }
}

if (child!=null) {
  try {
    String rojo = child.getStringProperty("red");
    String verde = child.getStringProperty("green");
    String azul = child.getStringProperty("blue");
    String negro = child.getStringProperty("black");
    String blanco = child.getStringProperty("white");
  }
  catch (NotExistInXMLEntity exception) {
```

```
    // mostrar el error y tomar alguna acción en respuesta
  }
}
```

Andami permite la creación de *tareas en segundo plano*, siendo posible además que estas tareas sean cancelables, es decir, interrumpibles. Si se desea realizar un procesamiento prolongado, es conveniente lanzarlo en un hilo independiente, esto se conoce como proceso en segundo plano. La finalidad es no bloquear la interfaz gráfica. Un exportación de una capa a un fichero que pueda ser costosa, es un ejemplo habitual de tarea lanzada en segundo plano. Las tareas en segundo plano se muestran como un panel donde se monitoriza el estado de la tarea, y un botón que permite la cancelación del proceso.

Para la creación de la tarea usaremos el método:

```
public static void cancelableBackgroundExecution IMonitorableTask task)
```

Para poder crear cómodamente una tarea en segundo plano, se utiliza *AbstractMonitorableTask* que implementa *IMonitorableTask* y que es posible extender anónimamente para crear dicha tarea:

```
PluginServices.cancelableBackgroundExecution(new AbstractMonitorableTask() {
  public void run() {
    for (int i=0; i<100001; i++) {
      System.out.println("hilo mundo "+i);
    }
  }
});
```

Con este código, la parte situada dentro del método run() se ejecutará en un hilo independiente. Para mostrar un diálogo de progreso y posibilitar la cancelación de la operación, es posible añadir algunas líneas:

```
PluginServices.cancelableBackgroundExecution(new AbstractMonitorableTask() {
  public void run() {
    setNote(PluginServices.getText(this, "Exportar_fichero"));
    setInitialStep(0);
    setFinalStep(100000);
    for (int i=0; i<100001; i++) {
      if (!isCanceled()){
        System.out.println("hilo mundo"+i);
        setCurrentStep(i);
      }
      else {
        return;
      }
    }
  }
});
```

Al llegar al *final step* final step, en el ejemplo 100.000) el diálogo desaparece automáticamente, y no desaparecerá hasta lleguar al final step pese a que el nuevo hilo ya haya terminado. Para necesidades especiales que *AbstractMonitorableTask* no cubre, hay varias opciones:

- Extender *AbstractMonitorableTask* de manera no anónima, y reescribir los métodos necesarios y el constructor.
- Extender *DefaultCancellableMonitorable* en una nueva clase que implemente a la vez *IMonitorableTask*, y reescriba los métodos necesarios y el constructor.
- Crear una clase nueva que implemente simplemente *IMonitorableTask* e incorpore toda la funcionalidad necesaria.

Al extender *AbstractMonitorableTask* de manera anónima y solicitar que se muestre el diálogo de progreso, se muestra un diálogo de progreso indeterminado, es decir, la barra que representa el progreso no avanza desde el inicio hasta el fin, si no que se mueve a izquierda y derecha continuamente hasta el final la tarea. A continuación se muestra un ejemplo en el que se extiende *AbstractMonitorableTask* de manera no anónima para mostrar una barra de progreso determinada:

```
public class BackgroundTask extends AbstractMonitorableTask {
  public BackgroundTask() {
    setDeterminatedProcess(true);
    setNote(PluginServices.getText(this, "Exportar_fichero"));
    setStatusMessage(PluginServices.getText(this, "Exportando..."));
    setInitialStep(0);
    setFinalStep(100000);
  }
  public void run() {
    for (int i=0; i<100001; i++) {
      if (!isCanceled()){
        System.out.println("hilo hilero"+i);
        setCurrentStep(i);

      }
      else {
        return;
      }
    }
  }
}
```

y la utilizaremos de la siguiente forma:

```
PluginServices.cancelableBackgroundExecution(new BackgroundTask());
```

Andami facilita la creación de *ficheros temporales*, que se trata de ficheros que se eliminan automáticamente al cierre de gvSIG, si gvSIG finaliza correctamente. Por un lado, dando nombre y unos datos, Andami crea el fichero temporal con el contenido que se le proporciona. Otro modo posible consiste en solicitar la ruta del directorio donde se ubican los ficheros temporales y crear en este directorio los ficheros necesarios. En los dos casos, los ficheros se eliminan al salir de gvSIG de forma automática. Para el acceso a este servicio no se utiliza *PluginServices* sino una clase llamada *Utilities*.

```
public static void createTemp(String fileName, String data)throws IOException
```

Crea un fichero con el nombre *fileName*, y escribe en él el contenido de *data*, finalmente cierra el fichero. Al salir de gvSIG, el fichero se eliminará automáticamente.

```
public static String createTempDirectory()
```

Crea el directorio para ubicar ficheros temporales, si no existe, y devuelve su ruta. Los ficheros creados en ese directorio serán eliminados automáticamente al cierre de gvSIG.

Andami ofrece la posibilidad de *acceder a recursos* como imágenes, cartografía, etc. del plugin. Es frecuente que los plugins incluyan diversos recursos que habitualmente son imágenes, algún fichero XML, cartografía en algún plugin especial, etc. Para acceder a estos recursos, no se debe utilizar ni rutas absolutas ni rutas relativas que incluyan el nombre del plugin, pues es poco portable, porque un cambio del nombre del plugin provocaría que dejase de funcionar. Lo correcto es el acceso a ellos utilizando una ruta relativa al directorio raíz del plugin. Cada plugin tiene un *PluginClassLoader* capaz de cargar recursos empleando rutas relativas al directorio raíz del plugin. Para acceder al *ClassLoader*, existen varias opciones:

```
PluginServices ps = PluginServices.getPluginServices(this);
PluginClassLoader loader = ps.getClassLoader();
```

o también

```
ClassLoader loader = this.getClass().getClassLoader();
```

Una vez obtenido el *ClassLoader*, se utiliza el método *getResource()* para obtener la URL de la ruta absoluta del recurso. Suponiendo, por ejemplo que se quiere cargar una imagen llamada close.png ubicada en

el subdirectorio images de un plugin. Es decir, la imagen está ubicada en gvSIG/extensiones/org.gvsig.elplugin/images/close.png y la ruta relativa al directorio raíz del plugin es images/close.png. Para el acceso a la imagen se usaría:

```
URL imageURL = loader.getResource("images/close.png");
if (imageURL!=null)
  ImageIcon icon = new ImageIcon(imageURL);
```

Andami posibilita el acceso al *portapapeles*. Permite la obtención y colocado de texto en el portapapeles.

```
public static void putInClipboard(String data)
```

Deposita en el portapapeles la cadena de texto *data*.

```
public static String getFromClipboard()
```

Obtiene el texto del portapapeles si existe, o null en caso contrario.

Estos métodos agilizan el acceso al portapapeles si se requieren manipulaciones simples, y consituyen simples atajos para los métodos estándar de Java . Si es necesario manipular otros tipos de datos, es posible usar las clases *ClipBoard*, *Transferable* y *DataFlavour* de Java y los métodos *Toolkit.getDefaultToolkit().getSystemClipboard()* y *Toolkit.getDefaultToolkit().getSystemSelection()*.

5.2.3 La librería libFMap

libFMap es un proyecto software desarrollado en Java que forma parte del Proyecto gvSIG, se trata de una librería que permite trabajar con información geográfica, cartográfica, topográfica, etc. según estándares internacionales, y al mismo tiempo con imágenes, en capas gráficas, tipo raster o vectorial. Con el paquete FMap se proporciona un control de interfaz de usuario y herramientas diseñadas para este control. Se trata del controlador de cartografía tanto para dibujar como para el acceso a los datos. Es el motor de SIG de la aplicación.

A grandes rasgos se compone de un gestor de herramientas, capas y orígenes de datos para éstas y las geometrías utilizaradas para la representación de los datos en las capas.

- *MapControl*: encargada de dibujar y mantener la herramienta actual, conoce todas las herramientas que existen en la aplicación

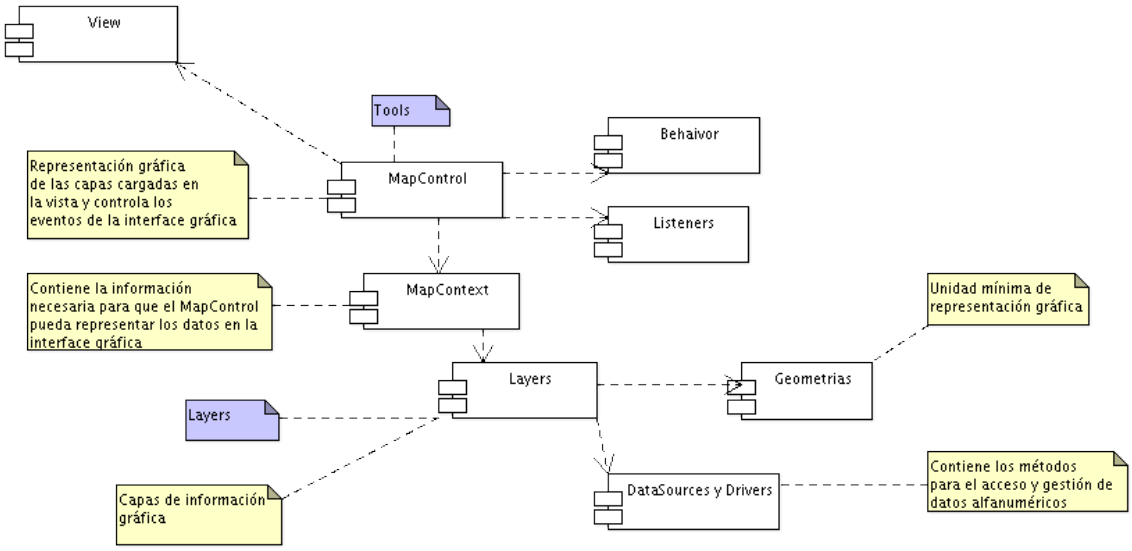


Figura 5.12: Diagrama de bloques de FMap.

- *MapContext*: Es el contexto de la parte gráfica. Tiene los elementos necesarios para que el MapControl pueda realizar su labor.
- *Behaivor*: Es un comportamiento de una herramienta. También dice como se comporta la herramienta gráficamente. Controla el dibujado de la herramienta y el iniciador de los eventos de la herramienta.
- *Listeners*: Son los encargados de gestionar los eventos de las distintas herramientas, ya sea propagándolos hacia quien corresponda o ejecutando las instrucciones necesarias.
- *Layer*: Contiene las características de la capa y las herramientas necesarias para su gestión.
- *Geometrías*: Son los distintos tipos de elementos gráficos que pueden ser representados dentro de una layer.
- *DataSources y drivers*: Contiene los métodos necesarios para la gestión de los datos tanto gráficos como alfanumericos.

Incluye drivers de acceso a los múltiples formatos, y dispone de un conjunto de herramientas para trabajar con las capas visuales: zoom, pan (arrastrar usando el ratón), selección de geometrías, etc.

Permite la edición de capas con un conjunto propio de herramientas. Trabaja con índices espaciales, lo que acelera el acceso a los datos (re-

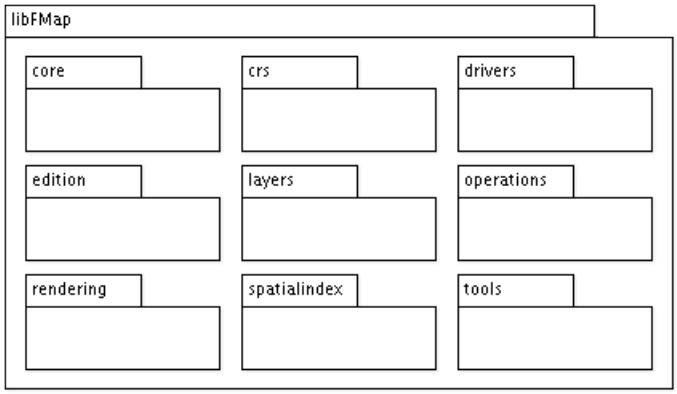


Figura 5.13: Visión estructural de la librería libFMap.

comendable si hay muchos en las capas, e incluso están en repartidos en múltiples niveles).

Existen una amplia variedad de tipos de capas, cada una con su propia estrategia para dibujarse y recorrer su información.

libFMap agrupa las capas en forma de árbol, de este modo se amplía el horizonte de trabajo, y puede servir tanto para obras civiles, medio ambiente, arquitectura, arqueología, etc.

5.2.3.1 MapControl

Se trata de un componente diseñado para la interacción con las distintas capas con información gráfica. Mediante herramientas el usuario puede interactuar con ellas. *MapControl* es un componente Java de intefaz gráfica. *MapControl* pinta un conjunto de capas con información gráfica, vía *MapContext* que las contiene, y captura eventos de ratón producidos en él vía un *Behavior* que define el comportamiento actual, enriqueciendo los eventos con la información necesaria que permita simular la herramienta de interacción actual, vía *ToolListener*, que complete la simulación.

gvSIG nos proporcionará una serie de herramientas, según el tipo de capas visibles y activas, con las que el usuario interacciona con el objeto *MapControl* que contiene las capas. Al seleccionar cualquier herramienta disponible, se está seleccionando un comportamiento para trabajar con *MapControl*, que pueden ser múltiples comportamien-

tos llamados cada uno *Behavior*, cada uno con una *ToolListener*. La *libFMap* define cada uno de estos comportientos, que procesan eventos de ratón producidos en *MapControl*, generando nuevos eventos con la información necesaria según su naturaleza. Estos nuevos eventos serán enviados a la herramienta actualmente seleccionada para interactuar con *MapControl*. Esta herramienta dispondrá de un icono que el usuario podrá visualizar en su cursor, y una serie de métodos serán invocados según el tipo de evento producido. Las *ToolListener* complementan a los *Behavior* para la simulación de una herramienta con la que interactuar con un objeto *MapControl*. En la librería *libFMap* se definen *ToolListener* básicas, aunque existen otras que heredan de éstas definidas en otros proyectos.

MapControl utiliza un doble-buffer para el dibujado, intentando, en la medida que sea posible, dibujar en él, y una vez finalizado, enviar esta información a pantalla. *MapControl* crea un objeto compartido de tipo *Cancellable*, mediante el cual notificará a *MapContext* y a las capas que estén dibujandose, que pueden seguir con el proceso de dibujado, o deben cancelarlo. Al utilizar una instacia de MapControl, se asignan sus posibles comportamientos, identificando cada uno con una cadena de texto. Posteriormente se le indicará al objeto *MapControl* que utilice como "herramienta activa" uno de esos comportamientos, según la herramienta con la que se esté trabajando. Se busca tener el menor tiempo de respuesta en la interacción con el usuario, por eso en caso que el proceso de pintado sea costoso, se actualiza la pantalla con el valor del buffer cada t milisegundos. *MapControl* atiende sus peticiones de pintado a traves de un objeto de tipo *Drawer2*. Éste notifica a su hilo trabajador que ejecute sólo una a la vez, manteniendo otra en espera. Si llegara una petición de pintado, mientras hay otra en espera, esta segunda se perdería. Si el hilo "trabajador" encargado de pintar, finalizare, se quedaría en espera pasiva, hasta una nueva notificación por parte de *Drawer2*.

El diagrama de la figura 5.14 muestra los principales elementos que intervienen con *MapControl*.

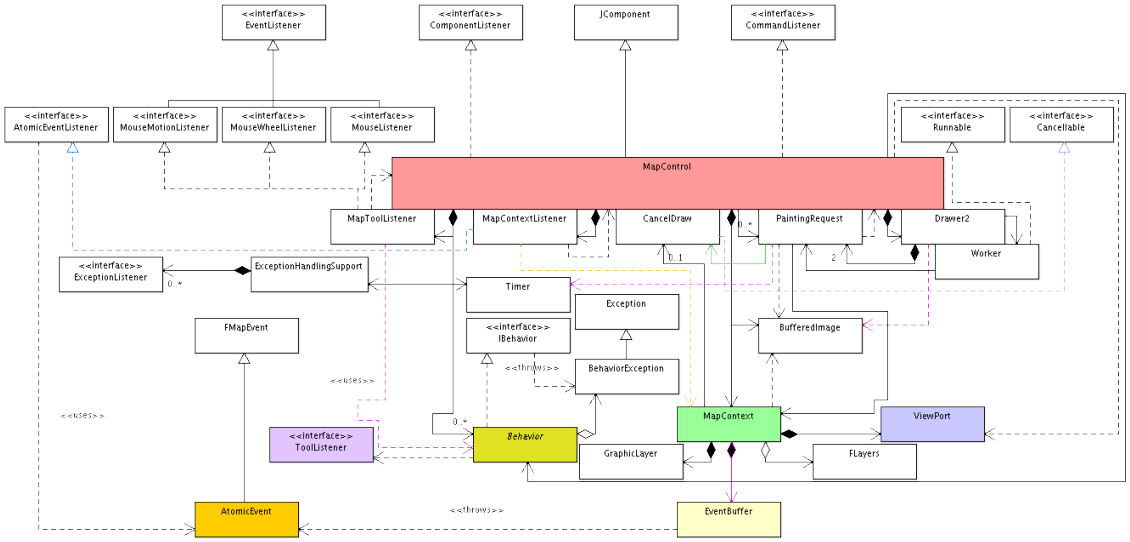


Figura 5.14: diagrama de clases simplificado de MapControl.

5.2.3.2 MapContext

Se trata de un modelo con cierto control y vista según el patrón Modelo-Vista-Controlador, utilizado por *MapControl* para almacenar, gestionar y visualizar capas con información gráfica. *MapContext* es una clase usada por *MapControl* para almacenar, gestionar y dibujar capas con información gráfica, y también los manejadores de eventos producidos en ellas.

Contiene un *ViewPort* que dispone de la información necesaria para la visualización de un área seleccionada de las capas en el área disponible para ello. Y contiene la conversión de las unidades definidas en el *ViewPort* a metros o a centímetros. *libFMap* permite trabajar con capas gráficas. Éstas se visualizarán de manera que se pueda trabajar con ellas mediante *MapControl*, que siguiendo el patrón Modelo-Vista-Controlador, se desentiende del almacenamiento o modelo, la gestión de eventos sobre ellas o parte del control, ni la transformación entre la parte que se quiere visualizar y la parte disponible para ello u otra parte del control de la que se encarga *ViewPort*, para encargarse solo de su visualización y la interacción con herramientas gráficas (vista, y parte del control).

De esta forma *MapContext* proporciona a *MapControl* la lógica necesaria para almacenamiento de capas, gestión de eventos en ellas y

su dibujado, para ello utilizando un puerto de vista *ViewPort* y una proyección. *MapContext* no puede existir fuera del contexto de *MapControl* . MapContext soporta dibujado de las capas que almacena y establece la calidad mediante antialiasing de texto, imágenes, y de renderizado, sin embargo la lógica de dibujado está contenida en cada capa.

El diagrama de la figura 5.15 muestra una visión en conjunto de las principales clases e interfaces que guardan relación con *MapContext*. MapContext implementa la funcionalidad definida en la interfaz *Projected* y es parte intrínseca de *MapControl*. Las relaciones con las capas que almacena *FLayers* y *GraphicLayer*, la información para dibujar el área seleccionada de las capas en el espacio disponible para esto (*ViewPort*), el buffer (*EventBuffer*) para el tratamiento de conjuntos de eventos recibidos de manera atómica, y una clase interna (*LayerEventListener*) para la manipulación genérica de los eventos en cualquier tipo de capa de éste. *GraphicLayer*, propia de MapContext para símbolos y geometrías editables, y un árbol con distinto tipo de capas gráficas. Éstas dos son opcionales, así que no es necesario que haya ambos tipos de capas a la vez. Finalmente el diagrama se completa con los tipos de excepciones que puede lanzar al trabajar con los eventos de las capas que contiene.

AtomicEvent muy útil para invocar listeners una vez realizadas una serie de operaciones, evitando así que se puedan invocar más veces, y que en el ejecución de alguna de éstas, se llegara a algún estado inestable. Con ello se evita además que puedan interferir o ralentizar el proceso de dibujado de capas, y también, mejorar la interactividad. Antes de recibir los eventos se debe activar el modo buffer en la instancia de *EventBuffer* invocando *beginAtomicEvent()*, y una vez se considere que ya no se van a recibir más eventos atómicos, se le debe de indicar mediante *endAtomicEvent()*. *AtomicEvent* es un tipo de evento propio de *libFMap*, este tipo de eventos se define genéricamente en la clase *FMapEvent*.

Los atributos de MapContext:

- Nodo raíz opcional de tipo FLayers para el árbol con las capas.

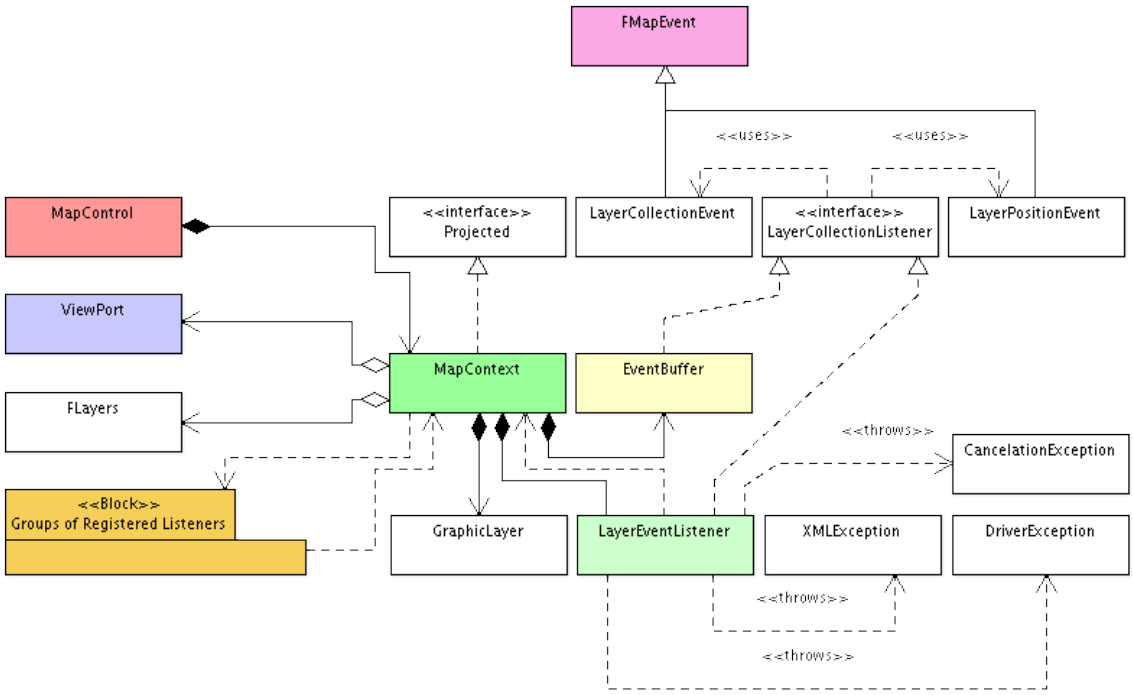


Figura 5.15: diagrama de clases de MapContex.

- Capa (*GraphicLayer*) opcional para geometrías y símbolos editables.
- *ViewPort* con información para que pueda dibujar MapControl la información seleccionada de las capas.
- Lista de listeners de tipo *LegendListener*, para los cambios en las leyendas de las capas.
- Lista de listeners de tipo *LayerDrawingListener*, para notificar que se ha pintado o se va a pintar una capa.
- Un buffer (*EventBuffer*) para almacenar eventos, y luego lanzarlos todos de una vez. Éstos pueden ser producidos por:
  - Una capa (*FLayer*): *LayerEvent*.
  - Una colección de capas (*FLayers*): *LayerCollectionEvent*, *LayerPositionEvent*.
  - Leyendas de una capa de tipo clasificable: *LegendChangedEvent*, *LegendEvent*.
  - El puerto de vista (*ViewPort*): *ExtentEvent*, *ColorEvent*, *ProjectionEvent*.
  - La selección en una capa alfanumérica: *SelectionEvent*.

  - Eventos atómicos: *AtomicEvent*.
- Un listener tipo *LayerEventListener*, para notificar eventos en cualquier tipo de capa contenida por *MapContext*. El tipo de eventos es:
  - Se ha añadido, eliminado o movido una capa del árbol de capas.
  - Una capa está a punto de añadirse, eliminarse o moverse del árbol de capas.
  - Ha cambiado la visibilidad de una capa.
  - Ha cambiado la selección de capas en el árbol.
- Una lista con los errores que se han producido en cualquier capa.
- Una lista con los errores que se han producido en el mapa.

La funcionalidad de MapContext:

- Atomicidad: iniciar o terminar un evento atómico usando *EventBuffer*, y agregar o eliminar listeners de tipo *AtomicEventListener* para manejar este tipo de eventos.
- Clonación: soporta dos modos, una clonación total, o una copia parcial con lo necesario para el pintado.
- Dibujado de capas: en principio, el dibujado que soporta es únicamente de las capas que almacena *MapContext*, con la lógica de dibujado de cada una. Para la aceleración del proceso de dibujado, únicamente se dibujan o redibujan aquellas capas en el estado sucias, por eso, *MapContext* soporta los siguientes tres tipos de dibujado:
  - Solo la capa con símbolos y geometrías.
  - Dibujado de todas las capas que lo requieran, es decir están en estado sucias.
  - Dibujado de todas las capas. Tanto en este caso como en el anterior puede establecer la calidad de dibujado mediante antialiasing de texto y símbolos, y también mediante la calidad del renderizado.
- Escala de la vista: obtener o establecer la escala del puerto de vista según la resolución en puntos por pulgada en la pantalla.

- Gestión de errores: agregar, obtener o borrar los mensajes de error producidos.
- Gestión de capas:
  - Obtener, establecer o dibujar la capa de tipo *GraphicLayer*.
  - Obtener las capas, asociar listeners o el buffer de eventos, o dibujarlas.
- Listeners: permite trabajar con distintos tipos de listeners:
  - *ErrorListener*: agregar este tipo de listener, o notificar a todos estos listeners registrados de un conjunto de eventos de error producidos durante una transacción atómica.
  - *LayerDrawingListener* y *LegendListener*: agregar, eliminar o invocar un listener de alguno de esos tipos.
- Persistencia: crear una nueva instancia de *MapContext* a partir de una entidad XML, o devolver una entidad XML que represente el objeto MapContext actual,con la menor información necesaria, esta entidad tendrá:
  - className: nombre completo de la clase.
  - 1 rama que será la entidad XML del *ViewPort* interno.
  - 1 rama que será la entidad XML a partir del nodo raíz del *FLayers* interno.
- Comparar con otro objeto *MapContext*: considerará igual al objeto si:
  - Ambos objetos son igual según el método equals de la clase *Object*.
  - Si tienen las mismas capas.
  - O, si tienen el mismo número de capas con el mismo nombre.
- Proyección: establecer y obtener la proyección que se utilizará con todas la capas que almacena *MapContext*.
- Puerto de vista: establecer u obtener el puerto de vista (ViewPort) actual de *MapContext*.
- Otra funcionalidad:
  - Crear un objeto *MapContext* a partir de un puerto de vista, y el árbol de capas actual.

- Obtener el extent ,que es la dimensión y posición del área, unión del de todas las capas.
- Obtener el extent seleccionado.
- Registar el *LayerEventListener* del *MapContext* a un nodo de tipo *FLayers* .
- Redibujado de todas las capas.
- Define el factor de zoom por defecto para alejar o acercar el encuadre del área visible:
  - Factor de Zoom In, para acercar: 2
  - Factor de Zoom Out, para alejar: 0.5

LayerEventListener se trata de una clase definida en *MapContext* para tratar eventos producidos en cualquier tipo de las capas de éste.
En concreto trata eventos de:

- Se ha añadido, movido o eliminado capa.
- Se está a punto de a añadir, mover o eliminar capa.
- Ha cambiado la visibilidad de alguna capa.

ViewPort es la clase utilizada por todas aquellas para realizar una transformación afín 2D entre 2 áreas rectangulares. Se obtienen coordenadas de píxel a partir de las de la capa visualo mapa. *MapContext* se encarga del almacenar la información necesaria para la visualización de capas en vistas de gvSIG. Un objeto de tipo MapContext contiene una instancia de *ViewPort* que le sirve para visualizar el área rectangular de trabajo seleccionada en las capas gráficas, en el área disponible, o imageSize, que hay para visualizarla en el *MapControl* que los contiene.

Hay 2 planos: el del mapa, y el de la pantalla. Cada uno puede estar en distinta unidad de medidad. *MapContext* permitirá realizar la conversión de medidas, entre ambos planos vía su *ViewPort*, teniendo en cuenta su proyección. *ViewPort* define varias unidades de medida como metro, kilómetro, milla, etc. En *ViewPort* al área seleccionada para la visualización del mapa, se conoce como extent, a la disponible para visualizarla, de la pantalla, *dimensionSize*, y a la seleccionada ajustada para cumplir la proporcionalidad en anchura y altura a la

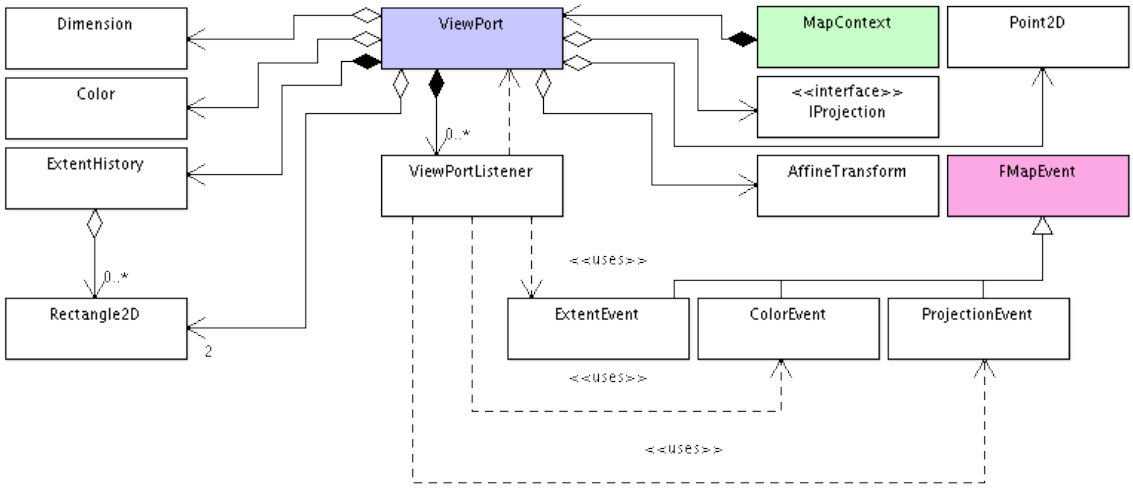


Figura 5.16: diagrama de clases simplificado de ViewPort en MapContext.

disponible, *adjustedExtent*. Al ajustar el área extra se rellenará con el mapa, o, en última instancia, con el color de fondo del *ViewPort*. *ViewPort* almacena en un objeto *ExtentHistory* los últimos extents, y da la opción de recargar el previo. Para acelerar el proceso de dibujado, se intenta pintar la menor información necesaria, así si al realizar la transformación mapa a ventana existen puntos muy cercanos que representan el mismo píxel o tres píxels juxtapuestos, únicamente se pintará uno. Por eso, cada vez que *ViewPort* recalcula la tranformación afín, además de recalcular también *adjustedExtent* y la escala entre este e *imageSize*, se calcula la distancia del mundo real a que equivalen 1 o 3 píxels juxtapuestos.

Atributos del ViewPort:

- Área seleccionada a visualizar en coordendas del mapa: *extent*.
- Dimensiones en pixels del área disponible para visualizarla: *imageSize*.
- Área para visualizar. Esta área es la seleccionada que se ajusta al aspecto de la disponible, de modo que ampliará la seleccionada, y en caso de no haber suficiente, con el color de fondo: *adjustedExtent*.
- Escala entre el área ajustada y la disponible. Esta escala es igual para el ancho y el alto.

- Color de fondo por defecto.
- Transformación afín entre el área a visualizar ajustada, en coordenadas de mapa, y la disponible para visualizarla, en coordenadas de píxel.
- Proyección que se utiliza para la obtención del área a visualizar.
- Unidad de medida de distancias en el área visualizada: *distanceUnits*.
- Unidad de medida en el mapa original: *mapUnits*.
- Lista de las últimas áreas visualizadas: *extents*. Permite situar vistas previas.
- Posición del área visible de la esquina donde empieza a visualizar el mapa: *offset*.
- Distancia en coordenadas del mundo que equivale a 1 pixel en la vista con el área a visualizar actual: *dist1pixel*.
- Distancia en coordenadas del mundo que equivale a 3 pixels en la vista con el área a visualizar actual: *dist3pixel*.
- Lista de listeners de tipo *ViewPortListener* asociados a este puerto de vista.

Funcionalidad de ViewPort:

- Áreas:
  - Área disponible: indicar y obtener las dimensiones, en pixels, del área disponible de visualización. Está en coordenadas de imagen.
  - Área seleccionada: indicar y obtener sus dimensiones y posición. En coordenadas del mundo, con su sistema de medida.
  - Área seleccionada ajustada a una escala de la disponible: permite la obtención de sus dimensiones y posición. En coordenadas del mundo, con su sistema de medida.
- Configuración:
  - Ajuste de Extent: permite adaptar el área a visualizar a una escala de la disponible antes de calcular la transformación afín.

- Listeners: lógica a ejecutar al producirse eventos sobre la vista. Permite agregar o eliminar listeners a la capa.
- Clonación: para obtener otra capa idéntica, independiente a la actual.
- Cálculo de la Transformación: se realiza automáticamente al refrescar el *ViewPort*, cambiar de extent, o cambiar de área disponible. Este proceso también recalcula la nueva área ajustada, la escala entre área ajustada y visible, y las distancias que equivalen en coordenadas del mundo a 1 y 3 pixels.
- Conversiones: *ViewPort* permite realizar conversiones entre datos (puntos, distancias o Rectangle2D (dimensión + posición de un área rectangular)) en coordenadas del mapa, y coordenadas de la imagen, o viceversa.
- Offset: indicar y obtener la posición donde empieza a situar el área seleccionada transformada en el área disponible.
- Persistencia: crear una nueva instancia de *ViewPort* partiendo de una entidad XML.
- Proyección: indicar y obtener la proyección usada para obtener el plano a partir de la información gráfica original, del que se visualiza un área.
- Unidades de medida: explicadas en el apartado Unidades de Medida.
- Otra funcionalidad:
  - Color de fondo: es posible cambiar y obtener el color de fondo por defecto.
  - Distancias equivalentes a 1 o 3 pixels: obtener la distancia equivalente en unidades de medida del mundo a 1 o 3 pixels en el área de visualización.
  - Obtener los extents previos.
  - Cambiar al extent previo, recalculando transformación afín, etc.
  - Refrescar, recalculando transformación afín.

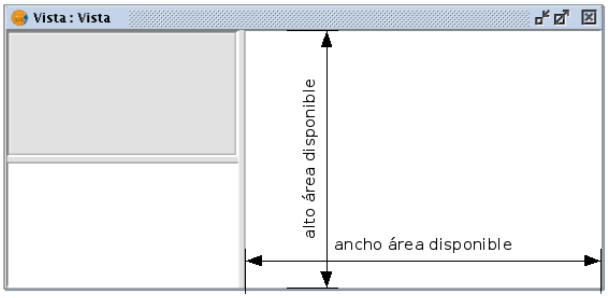


Figura 5.17: Una vista de gvSIG, donde se indica la dimensión del área disponible para visualizar las capas.

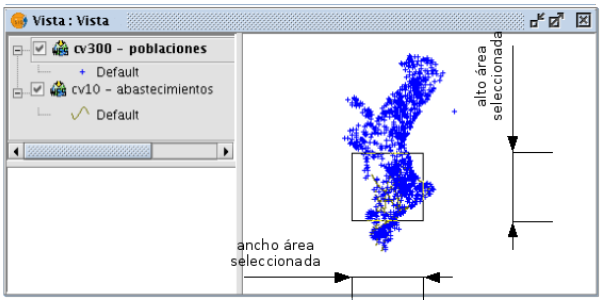


Figura 5.18: La vista de la Imagen 1, sobre la que se ha cargado dos capas. El usuario selecciona un área de interés, esta será el extent. Se recalculará el nuevo adjustedExtent.

- Cálculo de la distancia, en coordenadas del mundo, entre dos puntos de las capas gráficas originales. Se tiene en cuenta para ello la proyección con que se ha obtenido el plano a partir de dichas capas.

ViewPortListener es una interfaz para la captura y tratamiento de los eventos asociados a un *ViewPort*, que son:

- *ColorEvent*: cambio de color de fondo.
- *ExtentEvent*: cambio de área seleccionada.
- *ProjectionEvent*: cambio de proyección.

ExtentHistory sirve para tener un historial de Rectangle2D que representan áreas rectangulares seleccionadas de capas gráficas. Se gestionan de forma que se puede agregar nuevas, pero sólo almacena una cantidad máxima, por defecto 10, elimando las más antiguas, y obteniendo siempre la última agregada.

5.2.3.3 Layers

gvSIG permite trabajar con varias y diferentes tipos de capas, de forma que cada una, según su naturaleza, tiene unas propiedades y

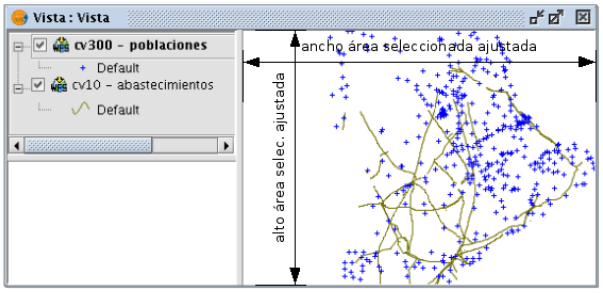


Figura 5.19: Área de interés seleccionada, adaptada a las dimensiones disponibles, ampliado el ancho seleccionado, por mantener una proporción al aspecto del área disponible de visualización.

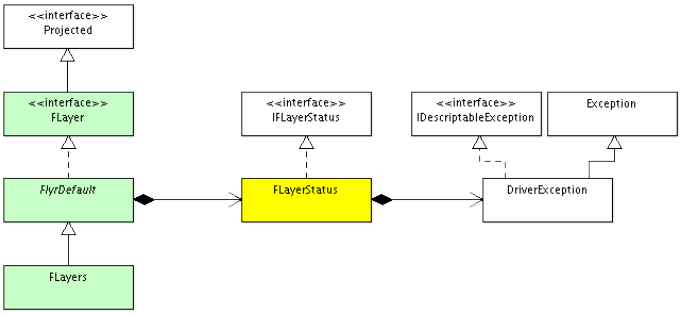


Figura 5.20: Diagrama de clases que muestra la relación de FLayerStatus con las capas de libFMap.

un compartamiento en ocasiones diferente. Sin embargo, todas las capas, teóricamente, pueden estar en un reducido número de estados simultáneos. A cada conjunto de estados simultáneos posible se le denomina estatus. Genéricamente, el estatus de una capa se define en la clase *FLayerStatus* de *libFMap*, ver el diagrama 5.20.

La clase abstracta de implementación común para todas las capas, *FLyrDefault*, contiene una instancia de tipo *FLayerStatus*, que representa el estatus actual. En la práctica, una capa únicamente podrá estar en un subconjunto de todos los posibles estatus, según su naturaleza.

El estatus almacena la información de las excepciones que se puedan ir produciendo al utilizar el driver asociado a ella, este tipo de excepción es *DriverException*.

Existen los siguientes estados:

- *Activa*: la capa está seleccionada en el TOC.
- *Visible*: la capa tiene seleccionado el checkbox asociado en el TOC. En caso afirmativo, se intentará pintar la capa a menos que no

(a) visibles, disponibles, y embalses activa.

(b) visibles, disponibles y ambas capas activas.

(c) no visibles, hidrografía activa, y ambas disponibles.

(d) visible embalses, y activa hidrografía.

(e) activa, no disponible y por tanto no visible.

(f) disponible, visible, activa, con permiso de escritura, y en edición.

Figura 5.21: Diferentes estados de las capas.

estén disponibles los datos necesarios, algo que sucede en ocasiones con capas de servicios remotos.

- *Disponible*: la fuente de datos de la capa está on-line, el driver ha conectado con ella con éxito y está preparada para que sus datos se puedan acceder sin problemas.
- Sucia: la capa requiere actualización. En el momento que una capa entra en edición, cambia la transparencia, cambia la visibilidad, se ordena su refresco, o se le indica por parte de un tipo de capa determinado, por ejemplo,cuando sus datos han cambiado, la capa pasará a estar sucia, hecho que obligará a ser repintada, para luego pasar a no sucia.
- *En TOC*: la capa se encuentra registrada en el árbol de capas del TOC.
- *Editándose*: puede modificarse el contenido de la capa en este momento.
- *Puede escribirse*: existe driver de escritura para esa capa, y tiene permisos para modificarla.

- *Caché de capas dibujadas*: almacena diversas imágenes para acelerar el pintado.

El status por defecto al crear una capa es: no activa, sí visible, sí disponible, no sucia, sí en TOC, no editándose, no puede ascribirse, no cachéde capas dibujadas.

Con "Layers" se engloba la definición genérica de las capas en gvSIG. *FLayer* dispone de la funcionalidad básica para trabajar con cualquier tipo de capa. *FlyrDefault*: capa genérica de la que heredan el resto. *FLayers*: conjunto genérico de capas. Toda capa en gvSIG hereda de *FlyrDefault*, que implementa la declaración genérica y común de capa, es decir el interfaz *FLayer* . *FlyrDefault*, es una clase abstracta, que dispone de métodos genéricos para el trabajo con capas definidos en la interfaz *FLayer*, métodos genéricos para el trabajo con drivers, para usar la capa, definidos en la interfaz *Driver*, y más métodos propios. *FLayer*, incluye métodos necesarios para el trabajo con proyecciones, definidos en la interfaz *Projected*, y otros métodos para trabajar con capas. Es posible tener una colección de capas como nodo de un árbol de capas, y de esta forma poder trabajar con distintos niveles. Para eso, se definió la clase *FLayers*, que hereda de *FlyrDefault*, e implementa la interfaz *LayerCollection* que dispone de métodos para trabajar con un conjunto de capas. Puesto que las capas de un mismo nodo pueden ser de distintos tipos, *FLayers* implementa, por compatibilidad, las interfaces *InfoByPoint* y *VectorialData*. Así, posibilitar que algunas capas trabajen con datos vectoriales, por ello se implementa la interfaz *VectorialData*, y al mismo tiempo es posible que soporten la operación de obtención de información a partir de un punto, por eso se implementa la interfaz *InfoByPoint*.

El diagrama de la figura 5.22 muestra lo explicado, sin entrar en detalles de métodos ni atributos, pero sí listeners que pueden asociarse a una capa, con los correspondientes eventos que pueden recibir, y tipos de excepciones que se pueden lanzar en caso error trabajando con una capa.

A continuación se describen los principales tipos de capas que se implementan en la librería libFMap o en otro proyecto de gvSIG heredando de *FLayer*, *FLyrDefault*, o, *FLayers*.

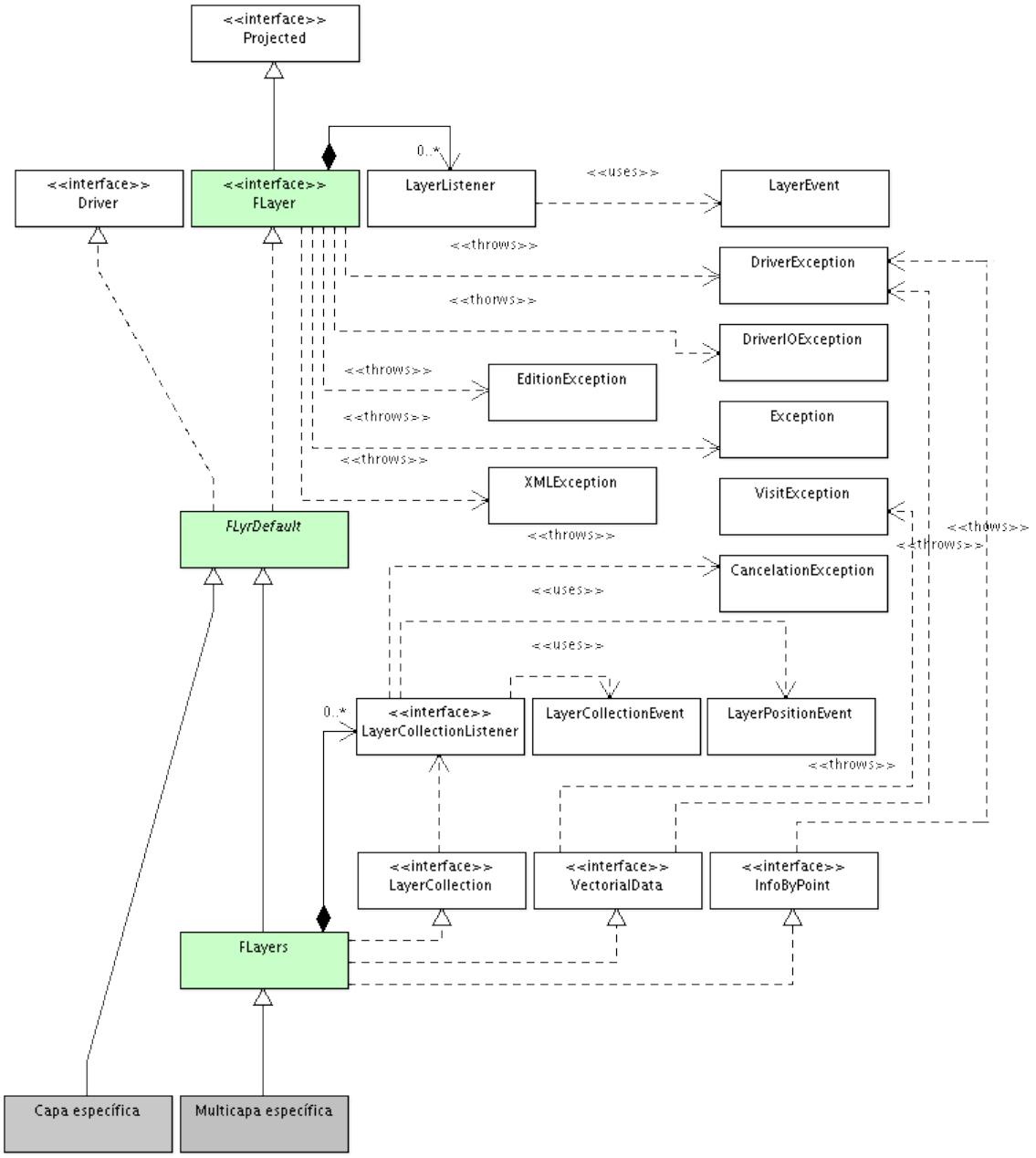


Figura 5.22: relación entre las clases e interfaces del grupo Layers.

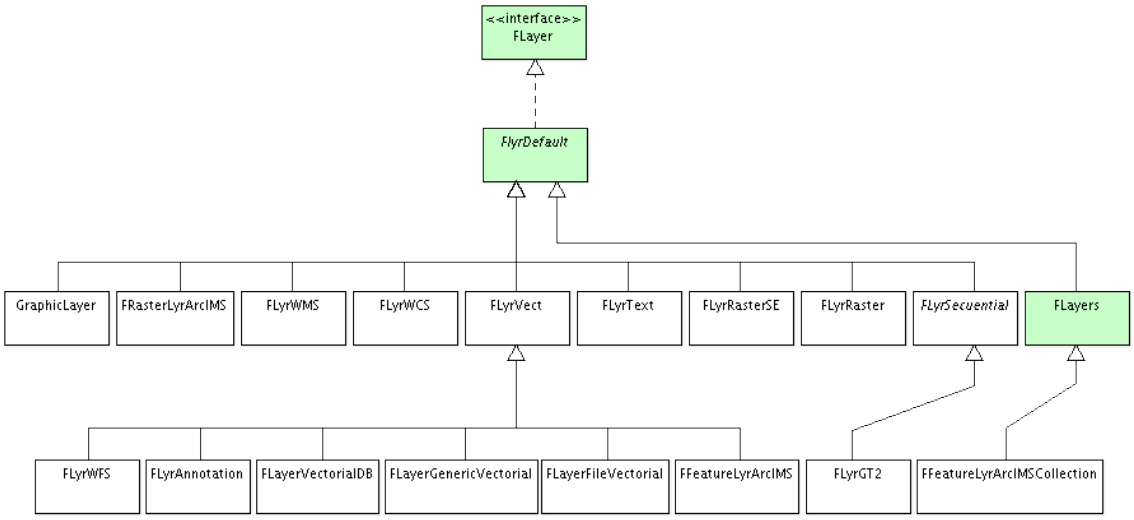


Figura 5.23: Diagrama simplificado que muestra los principales tipos de capas.

Entre los tipos más destacados para este proyecto se encuentran :

- *FLyrRaster* Antigua capa para el trabajo con imágenes ráster, que era ofrecida por libFMap. De tipo *FLyrDefault* reimplementada para dar soporte a operaciones propias de imágenes ráster. Permite también:
  - Indicar y obtener su estatus.
  - Agregar o eliminar ficheros a la capa ráster.
  - Indicar o conocer si será destruida la memoria del raster al eliminarlo del TOC.
  - Consultar si la imagen está o no georreferenciada.
  - Obtener el grid asociado.
  - Crear un buffer para tener un driver raster de memoria.
  - Indicar el extent, es decir, el área seleccionada de trabajo de la capa.
- *FLyrRasterSE*. Sustituye a la antigua *FlyrRaster*, y deja de estar en libFMap, para ubicarse en *extRasterTools-SE*. Aporta múltiples mejoras para dar soporte a gran cantidad de formatos raster existentes.
- *FLyrVect* Capa vectorial genérica: da soporte a edición, selección, reproyección, leyendas, índices vectoriales, datos alfanuméricos, acceso aleatorio a datos vectoriales, información por punto, e impresión. También se le puede establecer una estrategia de recorri-

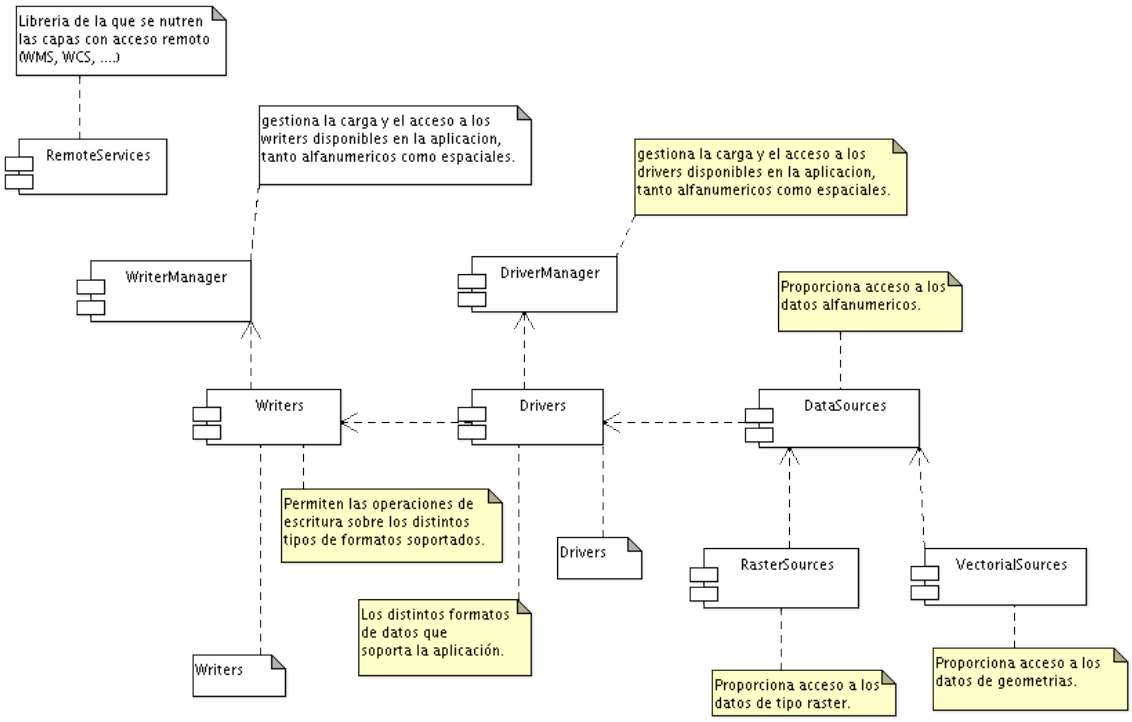


Figura 5.24: Diagrama de bloques de Subdriver.

do, manipulación y pintado de sus features, y permite consultar por área rectangular, punto y shape. También realiza un "marshall" de su información, para poder ser salvarda en un fichero ".gvp"de gvSIG y así restaurarla posteriormente.

5.2.3.4 DataSources y Drivers

Contiene las clases y métodos que utiliza FMap para la gestión de los datos.

Es el bloque que está recuperando los datos directamente de la fuente. Sirve de puente entre la aplicación y los datos. Contiene las clases necesarias para acceder a los datos, escribir datos en una fuente, así como las propiedades de acceso a fuentes remotas.

RemoteServices: Contiene las herramientas para unificar el acceso a datos remotos.

- *Drivers*: Gestionan los distintos tipos de datos que soporta gvSIG.
- *DriverManager*: Proporciona la carga y el acceso a los drivers disponibles en la aplicación alfanuméricos y espaciales.

- *WriterManager*: Proporciona la carga y el acceso a los writers disponibles en la aplicación alfanuméricos y espaciales.
- *Writers*: Permiten las operaciones de escritura sobre los formatos soportados.
- *VectorialSources*: Proporciona acceso a los datos con las geometrías.
- *DataSources*: Proporciona acceso a los datos alfanuméricos.
- *RasterSources*: Proporciona acceso a los datos de tipo ráster.

5.3 Análisis funcional

5.3.1 Introducción

El propósito del presente apartado es definir los requerimientos que debe tener la aplicación. Como el nombre del proyecto indica, abordará la gestión de un proyecto LiDAR en un sistema de información geográfica, en nuestro caso en gvSIG. El propósito de la especificación de requerimientos es formalizar funcionalidades de la aplicación junto al cliente. En nuestro caso concreto al tratarse de un desarrollo interno de la empresa no existe tal cliente, pero no por ello se dará menos importancia a este apartado pues para producir un software de calidad debe haber concordancia entre los requisitos funcionales y de rendimiento establecidos explícitamente.

5.3.2 Casos de uso

Los casos de uso describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista de los usuarios que van a utilizarlo, es decir, son descripciones de la funcionalidad del futuro sistema independientemente de la implementación. En este documento se incluyen diagramas como representación de los casos de uso. Un diagrama es una descripción completa de un sistema desde una perspectiva concreta. Su notación permite la comprensión de aquello que se está desarrollando como producto y que representa los requisitos funcionales de éste. Cada caso de uso está formado por una secuencia de eventos, iniciada por un actor, que describe la interacción que tiene lugar entre el actor y el sistema. En nuestro caso el actor será el técnico LiDAR. Los diagramas de casos de uso son fácilmente

compresibles tanto por clientes o usuarios. Representan los requisitos funcionales del sistema. Se utilizan como base para un desarrollo iterativo e incremental.

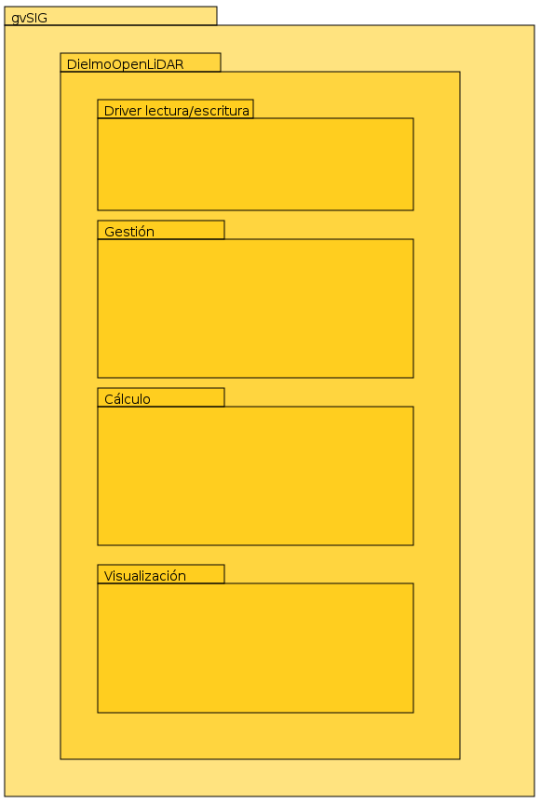


Figura 5.25: Diagrama inicial de subsistemas.

En el diagrama inicial ver imagen 5.25 se representa la agrupación gerárquica de los distintos casos de uso. En primer lugar en la figura 5.26 se muestra el diagrama de casos de uso del subsistema de gestión. En el subsistema de gestión se distinguen 4 funcionalidases relativas a la gestión de un proyecto LiDAR. En primer lugar la creación de un proyecto LiDAR es el primer paso en la inicialización de un proyecto. Se le asigna un nombre por defecto que es LiDAR - [numero de proyecto], una fecha de creación correspondiente a la fecha actual. También es posible definir el propietario y el operador del proyecto LiDAR. En segundo lugar el caso de uso definir un proyecto LiDAR. Este caso de uso engloba toda la funcionalidad mediante la cual el técnico LiDAR determina cuales son los datos de entrada del proyecto. Para una mayor comprensión de éste aparece desglosado en los distintos casos de uso que lo conforman. En la figura 5.27 se muestra el ca-

so de uso definir un proyecto LiDAR. En tercer lugar el caso de uso modificar un proyecto LiDAR representa la acción de cambiar ciertos parámetros que han sido definidos en un proyecto LiDAR ya creado. En cuarto y último lugar en el subsistema de gestión se encuentra el caso de uso guardar proyecto LiDAR. Este caso de uso representa la funcionalidad de persistir el proyecto LiDAR. La persistencia en este caso consiste en guardar de manera integrada al proyecto gvSIG sobre el que trabajamos todos los datos adicionales que el proyecto LiDAR ha incorporado al sistema. Para conseguirlo se han incluido en la estructura del archivo gvp [2] todos estos datos de manera que sea posible recuperar el el estado del proyecto de gvSIG que contenga proyectos LiDAR definidos.

El caso de uso definir proyecto LiDAR merece un mayor detalle. En la figura 5.27 se muestra el diagrama de caso de uso completo definir un proyecto LiDAR. Además se incluyen las tablas de descripción con los casos de uso que lo conforman. Concretamente: definir datos LiDAR (figura 5.28), definir linea de vuelo, definir zonas de control (figura 5.29), definir área de interés (figura 5.32) , definir cartografía ráster (figura 5.30), definir cartografía vectorial (figura 5.31), definir leyenda (figura 5.39), definir shape de bloques (figura 5.33), definir ruta (figura 5.34), definir resolución (figura 5.35), definir solape (figura 5.36), tamaño máximo de edificio (figura 5.37), definir productos (figura 5.38), definir leyenda de clasificación (figura 5.39), definir operador (figura 5.40), definir fichero log (figura 5.41).

Por otro lado existen varios formatos de datos LiDAR. De ahi la pretensión de implementar herramientas de conversión entre ellos para completar funcionalida en el proyecto.Los ficheros .XYZ son ficheros de texto. En cada linea del fichero se encuentran los datos de un punto separados por un separador que puede ser un espacio, una coma, o un tabulador. Puede contener los campos X, Y, Z pudiendo incorporar el campo de intensidad. Se trata de un formato con el que se trabajaba antes de la aparición de formatos binarios como el .las. El formato .las se trata de un archivo binario alternativo a los sistemas propietarios o un archivo ASCII genérico sistema utilizado por muchas empresas. Un

[2] Archivo xml que contine toda la información de un proyecto de gvSIG al ser persistido. En él ha sido integrada toda la información del proyecto LiDAR para la persistencia de los datos que conforman el proyecto.

problema con los sistemas propietarios es que los datos no pueden ser fácilmente tomados de un sistema o proceso. Además, el rendimiento de procesamiento se degrada porque la lectura e interpretación de datos de elevación ASCII puede ser muy lento y el tamaño del archivo puede ser muy grande, incluso para pequeñas cantidades de datos. Esto puede ser una barrera significativa para la primera vez que los usuarios usan los datos lidar. Otro problema es que todos los datos en bruto e información específica para la recogida de datos LIDAR se pierde. Esto puede inhibir la solución de problemas y la depuración de los datos de los boletines de problemas y limitar el análisis a terceros de integridad de datos. El archivo con formato.LAS está pensado para atender muchas de estas cuestiones. Es un formato de archivo binario que mantiene información específica a la naturaleza de los datos LIDAR si bien no es excesivamente complejo. Por lo tanto pese a que sabemos que el formato .LAS es deseable frente al .XYZ, existen datos antiguos en éste último formato que no se desean perder, y es por tanto de gran utilidad proveer de estas herramientas de conversión. El formato .bin es un formato propietario de Terrasolid. Igual que los .LAS se trata de un formato binario.

Caso de uso	**crear proyecto LiDAR**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea iniciar un proyecto de datos LiDAR y quiere crear un nuevo proyecto en gvSIG para tal fin.
Precondiciones	El técnico ha iniciado el sistema.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
El técnico selecciona el tipo de documento Documento LiDAR en el gestor de proyectos de gvSIG y pulsa en el botón nuevo. A continuación puede cambiar el nombre que por defecto el sistema le ha dado pulsando en renombrar.	Se crea un nuevo documento LiDAR que aparece listado en el propio gestor de proyectos de gvSIG.

Tabla 5.1: Caso de uso crear proyecto LiDAR.

Caso de uso	**Definir datos LiDAR**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué ficheros de datos LiDAR formaran parte del proyecto LiDAR sobre el que trabaja.
Precondiciones	Se ha tenido que crear un proyecto de tipo LiDAR sobre el que se definen los datos.
Postcondiciones	
Incluye	
Extiende	Definir datos LiDAR ajustados: ficheros que forman parte de los datos LiDAR del proyecto y ya se encuentran ajustados. Definir datos LiDAR sin ajustar: ficheros que forman parte de los datos LiDAR del proyecto y no se encuentran ajustados.
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción datos subopción LiDAR. Añade los ficheros correspondientes a los datos LiDAR ajustados y/o añade los correspondientes a los no ajustados.	

Pulsa aceptar	El sistema recoge la información introducida y calcula el bounds2D de cada fichero definido. Este dato es necesario para el funcionamiento de la herramienta de cargado rápida de capas mediante intersección con un rectángulo sobre la vista.

Tabla 5.2: Caso de uso definir datos LiDAR.

Caso de uso	**Definir línea de vuelo**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué fichero vectorial contiene la línea de vuelo del proyecto con el que trabaja.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *datos* subopción *LiDAR*. Añade el fichero correspondientes a la línea de vuelo. Pulsa aceptar	El sistema recoge la información introducida y calcula el bounds2D del fichero definido. Este dato es necesario para el funcionamiento de la herramienta de cargado rápida de capas mediante intersección con un rectángulo sobre la vista.

Tabla 5.3: Caso de uso definir línea de vuelo.

Caso de uso	**Definir zonas de control**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué fichero vectorial de puntos que contienen los levantamientos GPS que se tomarán como zonas de control del proyecto con el que trabaja.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *datos* subopción *zonas de control*. Añade el fichero tipo shape de puntos correspondiente a las zonas de control vectorial. Pulsa aceptar	El sistema recoge la información introducida y calcula el bounds2D del fichero definido. Este dato es necesario para el funcionamiento de la herramienta de cargado rápida de capas mediante intersección con un rectángulo sobre la vista.

Tabla 5.4: Caso de uso definir zonas de control.

Caso de uso	**Definir campo de altura**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué campo es el correspondiente a la altura en el fichero vectorial de puntos que contienen los levantamientos GPS que se ha definido como zonas de control del proyecto con el que trabaja.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada. Se ha definido un fichero shape de puntos que contiene las zonas de control.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *datos* subopción *zonas de control*. Selecciona del desplegable el campo correspondiente a la altura en el fichero definido como zonas de control.	El sistema recoge la información introducida.

Tabla 5.5: Caso de uso definir campo de altura.

Caso de uso	**Definir cartografía ráster**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué ficheros de cartografía ráster formaran parte del proyecto LiDAR sobre el que trabaja.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *datos* subopción *ortofotos*. Añade los ficheros correspondientes a las ortofotos que forman parte del proyecto. Pulsa aceptar	El sistema recoge la información introducida y calcula el bounds2D de cada fichero definido. Este dato es necesario para el funcionamiento de la herramienta de cargado rápida de capas mediante intersección con un rectángulo sobre la vista.

Tabla 5.6: Caso de uso definir cartografía ráster.

Caso de uso	**Definir cartografía vectorial**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué ficheros de cartografía vectorial formaran parte del proyecto LiDAR sobre el que trabaja.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *datos* subopción *vectoriales*. Añade los ficheros correspondientes a las capas vectoriales que forman parte del proyecto. Pulsa aceptar	El sistema recoge la información introducida y calcula el bounds2D de cada fichero definido. Este dato es necesario para el funcionamiento de la herramienta de cargado rápida de capas mediante intersección con un rectángulo sobre la vista.

Tabla 5.7: Caso de uso definir cartografía vectorial.

Caso de uso	**Definir leyenda a cartografía vectorial**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué fichero de leyenda gvl asigna a una capa vectorial previamente definida .
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada. Se ha añadido alguna capa vectorial.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *datos* subopción *vectoriales*. Añade el fichero correspondiente a la leyenda pulsando el botón *Asignar leyenda* teniendo seleccionada la capa vectorial a la que se le pretende asignar ésta. Pulsa *aceptar*	El sistema recoge la información introducida.

Tabla 5.8: Caso de uso definir leyenda a cartografía vectorial.

Caso de uso	**Definir area de interés**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué fichero vectorial contiene el área de interés del proyecto LiDAR sobre el que trabaja.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *parámetros*. Añade el fichero vectorial correspondiente al área de interés que delimita la zona caso de estudio del proyecto . Pulsa aceptar	El sistema recoge la información introducida y calcula el bounds2D de cada fichero definido. Este dato es necesario para el funcionamiento de la herramienta de cargado rápida de capas mediante intersección con un rectángulo sobre la vista.

Tabla 5.9: Caso de uso definir area de interés.

Caso de uso	**Definir shape de bloques**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar qué fichero vectorial contiene el shape de bloques del proyecto LiDAR sobre el que trabaja.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *parámetros*. Añade el fichero vectorial correspondiente al shape de bloques. Se trata de un fichero que contiene parcelada el área de interés en bloques de un tamaño determinado que será utilizado como entrada para algoritmos de cálculo de manera que permita un procesado de los datos en porciones pequeñas.	

Pulsa aceptar	El sistema recoge la información introducida y calcula el bounds2D de cada fichero definido. Este dato es necesario para el funcionamiento de la herramienta de cargado rápida de capas mediante intersección con un rectángulo sobre la vista.

Tabla 5.10: Caso de uso definir shape de bloques.

Caso de uso	**Definir ruta**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar el directorio de trabajo.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *parámetros*. Añade la ruta al directorio de trabajo del proyecto. En éste se guardarán los ficheros auxiliares y productos finales derivados de cálculos que se realicen. Pulsa aceptar	El sistema recoge la información introducida.

Tabla 5.11: Caso de uso definir ruta.

Caso de uso	**Definir resolución**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar la resolución de trabajo.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *parámetros*. Añade la reosución (en metros) de trabajo del proyecto. Pulsa aceptar	 El sistema recoge la información introducida.

Tabla 5.12: Caso de uso definir resolución.

Caso de uso	**Definir solape**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar el solape.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *parámetros*. Añade el solape (en metros) de trabajo del proyecto. Pulsa aceptar	El sistema recoge la información introducida.

Tabla 5.13: Caso de uso definir solape.

Caso de uso	**Definir tamaño máximo de edificio**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar el tamaño máximo de edificio del proyecto actual.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *parámetros*. Añade el tamaño máximo de edificio (en metros) del proyecto.	
Pulsa aceptar	El sistema recoge la información introducida.

Tabla 5.14: Caso de uso definir tamaño máximo de edificio.

Caso de uso	**Definir productos principales**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar los productos principales que se desean obtener.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *parámetros*. DIELMO ofrecerá una lista de productos finales disponibles en futuras versiones del proyecto. El técnico selecciona los que interesan para el proyecto. Pulsa aceptar	El sistema recoge la información introducida.

Tabla 5.15: Caso de uso definir productos principales.

Caso de uso	**Definir productos auxiliares**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar los productos finales que se desean obtener.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *parámetros*. Pulsa aceptar	 El sistema recoge la información introducida.

Tabla 5.16: Caso de uso definir productos auxiliares.

Caso de uso	**Definir clasificación**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar la simbología de clasificación de los datos LiDAR.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *clasificación.* Añade y/o elimina cuantos nuevos símbolos desee asignándole color, valor, nombre y visibilidad. Pulsa aceptar	El sistema recoge la información introducida.

Tabla 5.17: Caso de uso definir clasificación.

Caso de uso	**Definir operador**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar su nombre.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *info*. Añade su nombre en el campo correspondiente. Pulsa aceptar	 El sistema recoge la información introducida.

Tabla 5.18: Caso de uso definir operador.

Caso de uso	**Definir fichero log**
Actores	técnico LiDAR
Descripción	El técnico LiDAR desea indicar el fichero log.
Precondiciones	Existe un proyecto LiDAR ya creado sobre el que se definen los datos de entrada.
Postcondiciones	
Incluye	
Extiende	
Hereda de	
Intenciones de usuario	Obligaciones del sistema
Abre el proyecto LiDAR sobre el que quiere definir los datos. Selecciona en el menú la opción *info*. Añade el fichero log donde se registrarán las operaciones que hayan sido realizadas en el proyecto	
Pulsa aceptar	El sistema recoge la información introducida.

Tabla 5.19: Caso de uso definir fichero log.

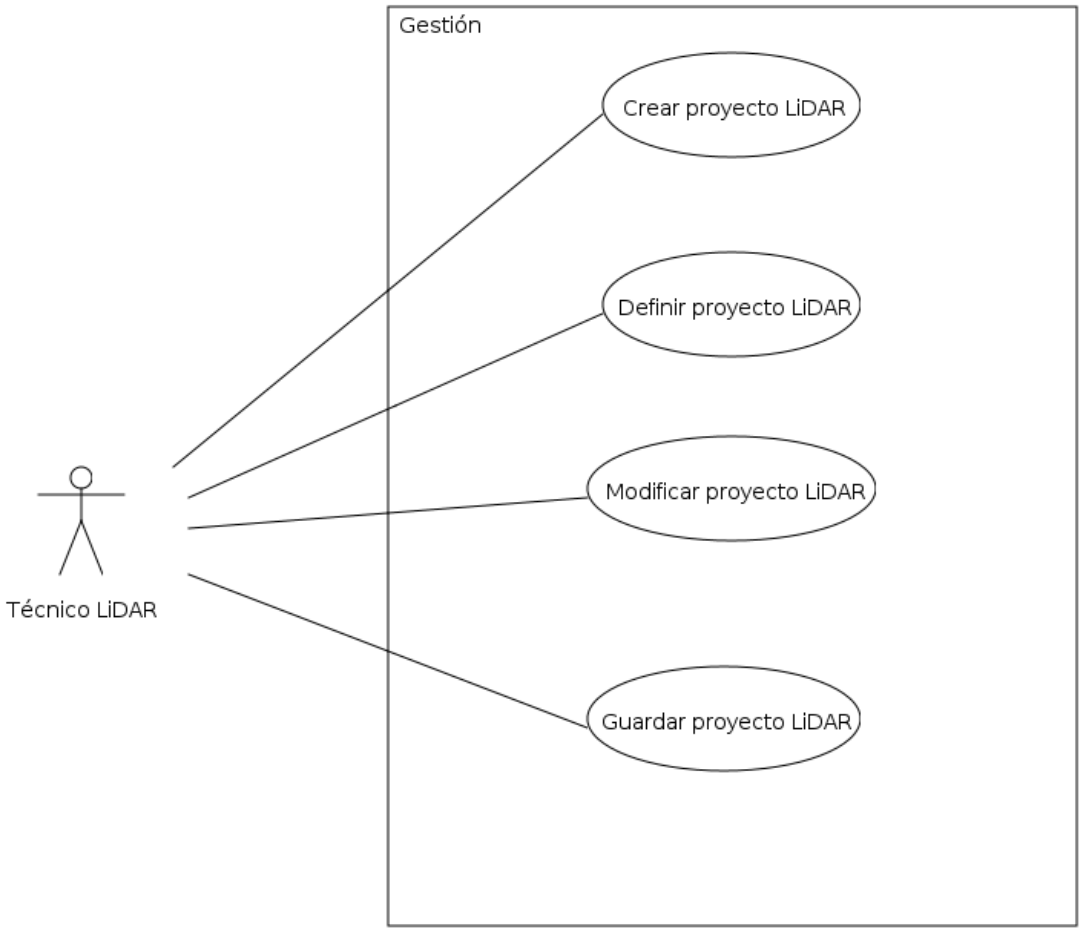


Figura 5.26: Diagrama de contexto del subsistema gestión.

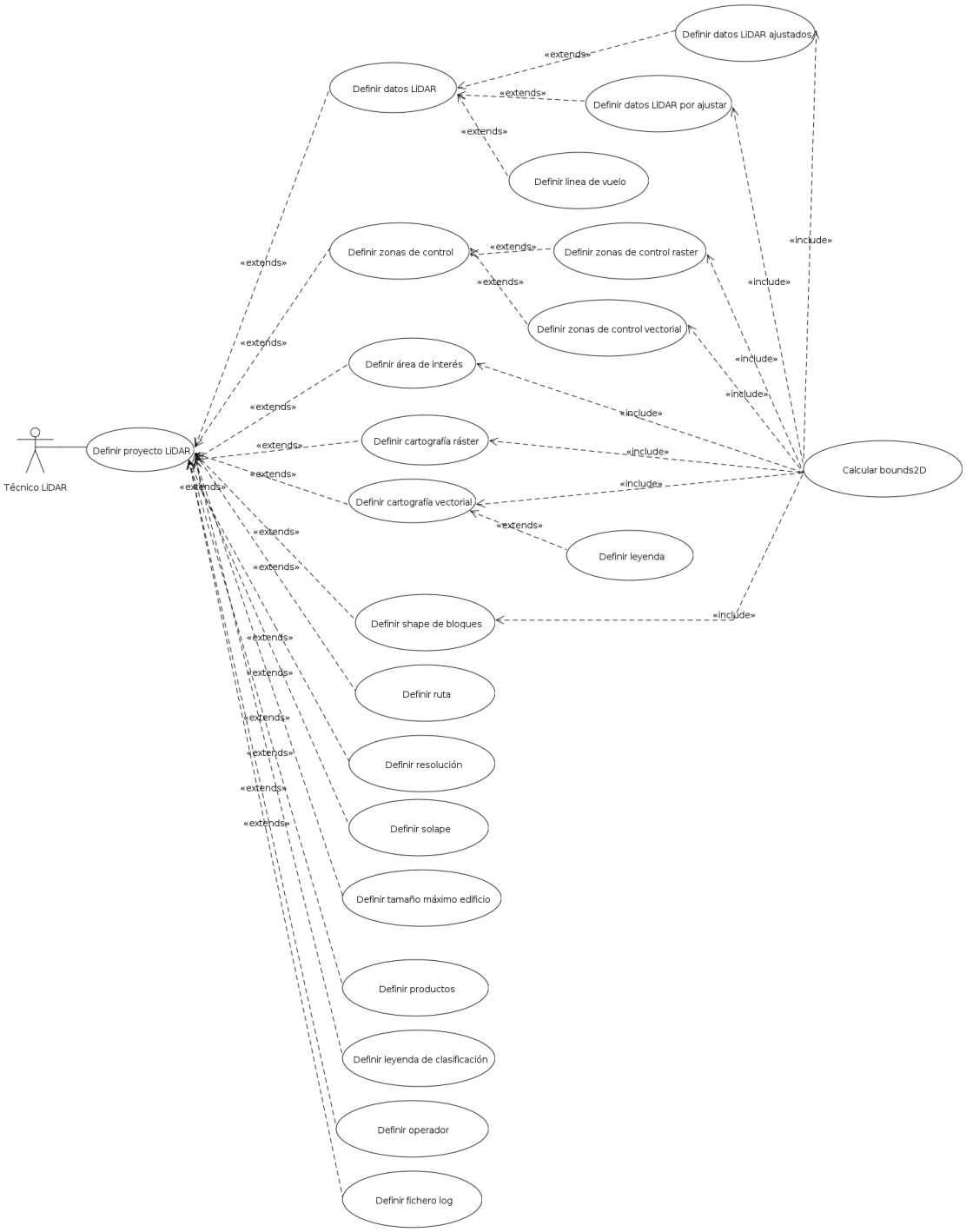


Figura 5.27: Diagrama de contexto del caso de uso definición de un proyecto LiDAR.

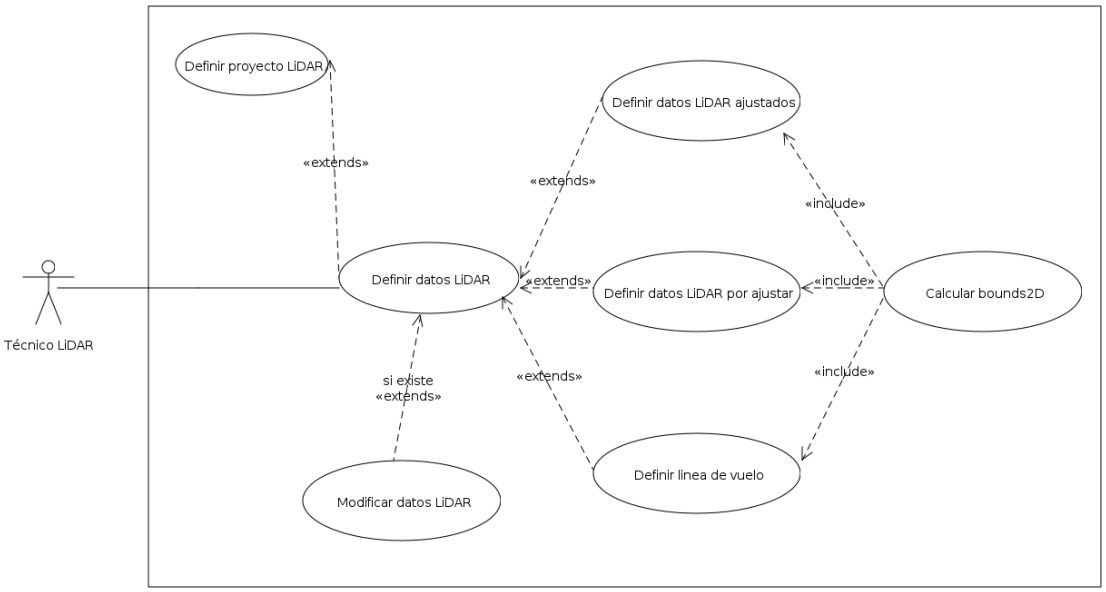


Figura 5.28: Diagrama de contexto del caso de uso definir datos LiDAR.

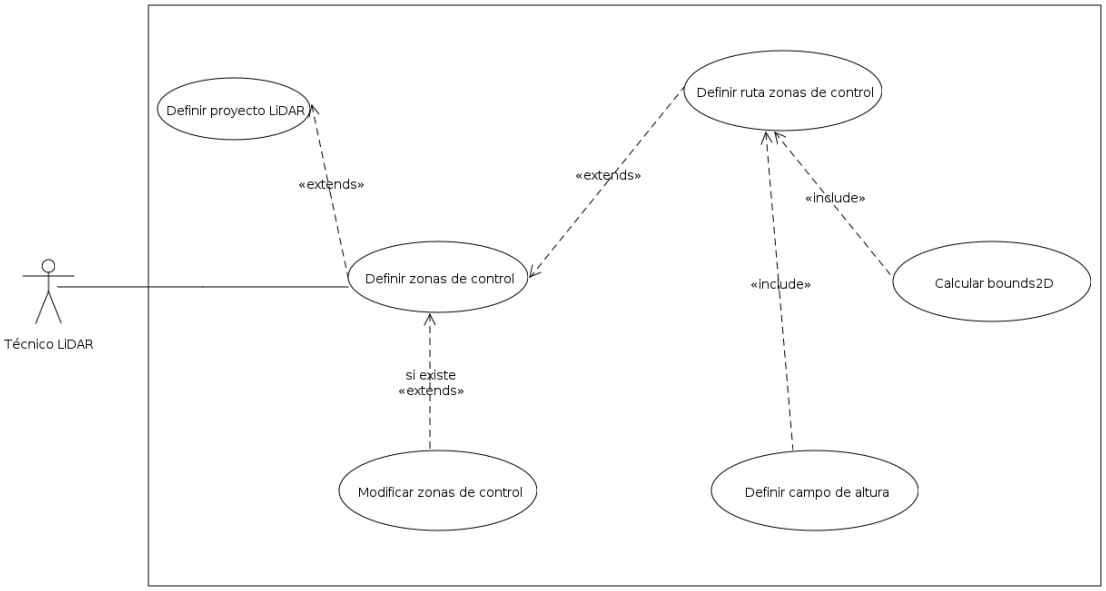


Figura 5.29: Diagrama de contexto del caso de uso definir zonas de control.

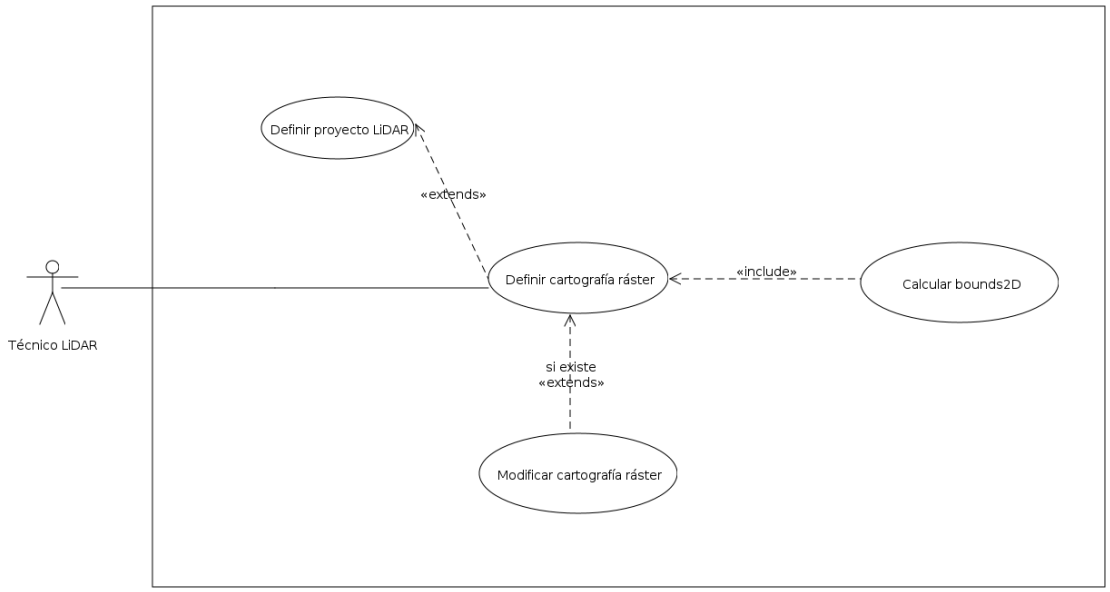


Figura 5.30: Diagrama de contexto del caso de uso definir cartografía ráster.

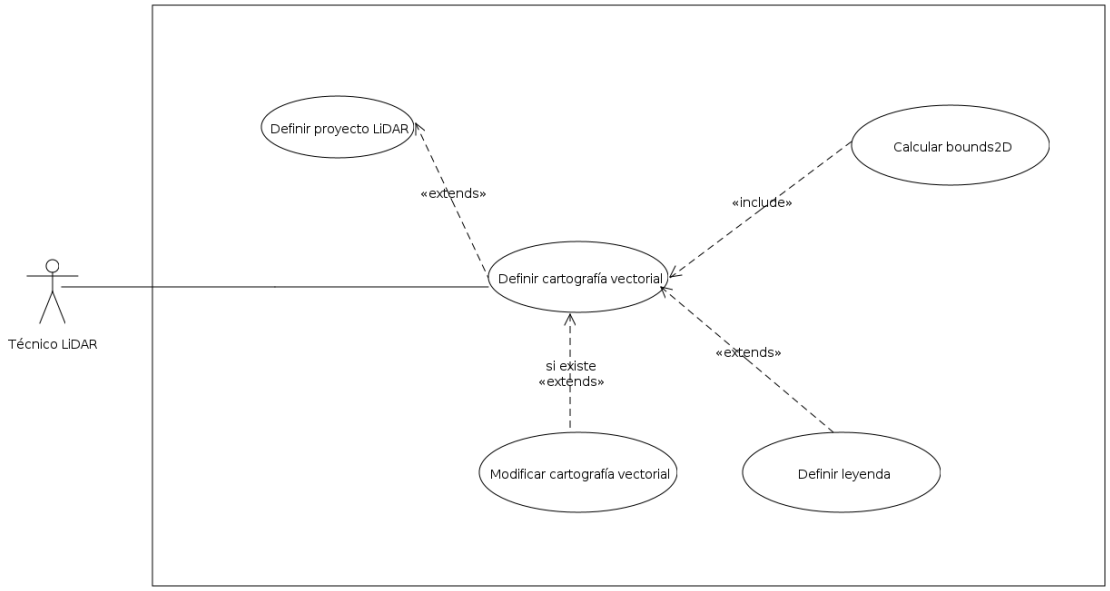


Figura 5.31: Diagrama de contexto del caso de uso definir cartografía vectorial.

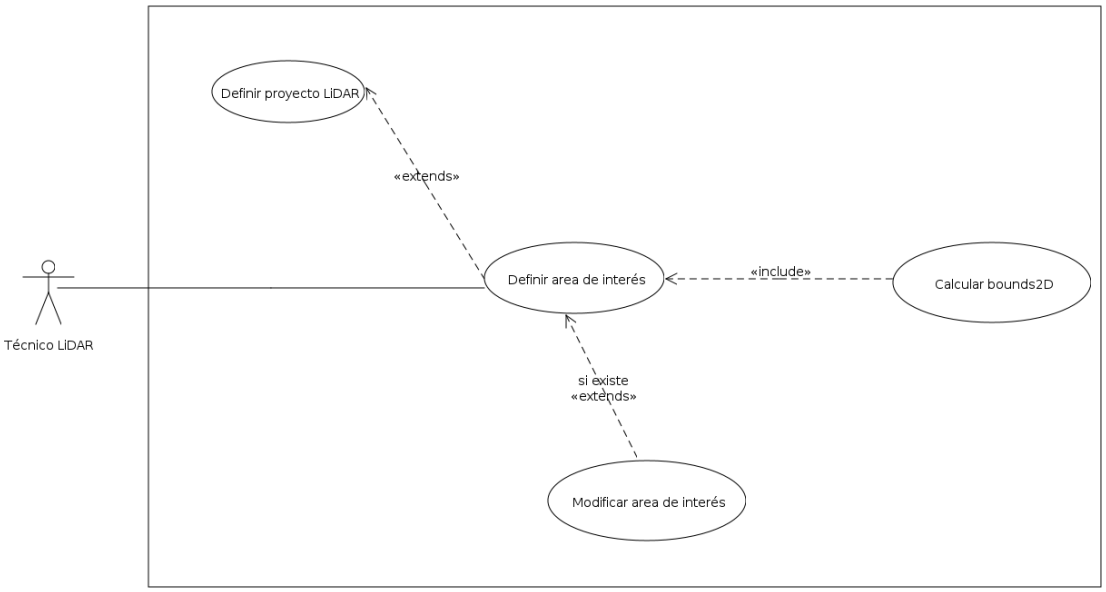


Figura 5.32: Diagrama de contexto del caso de uso definir area de interés.

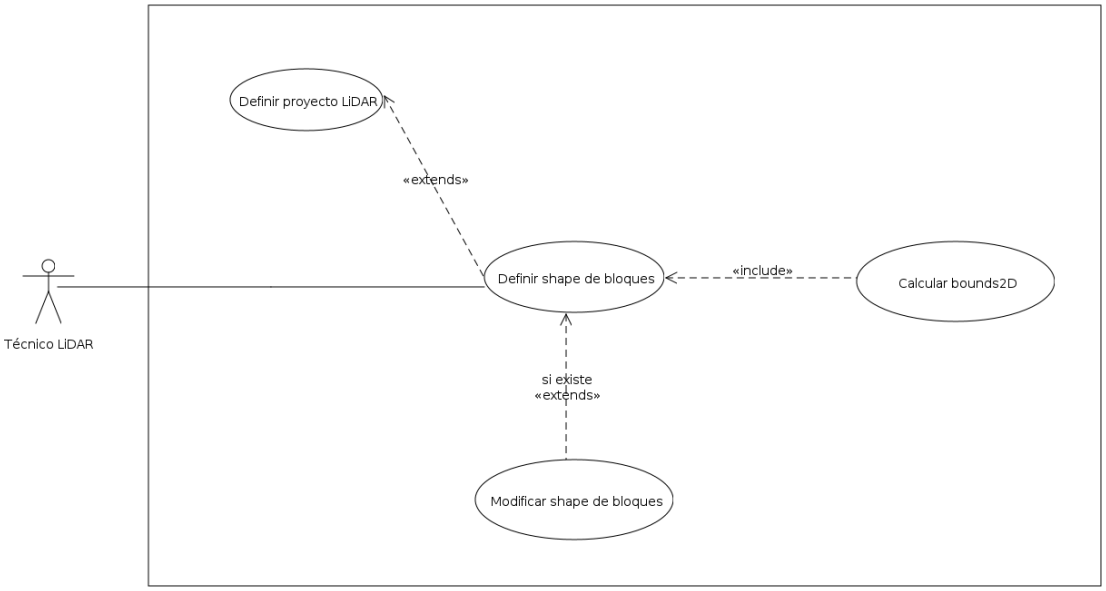


Figura 5.33: Diagrama de contexto del caso de uso definir shape de bloques.

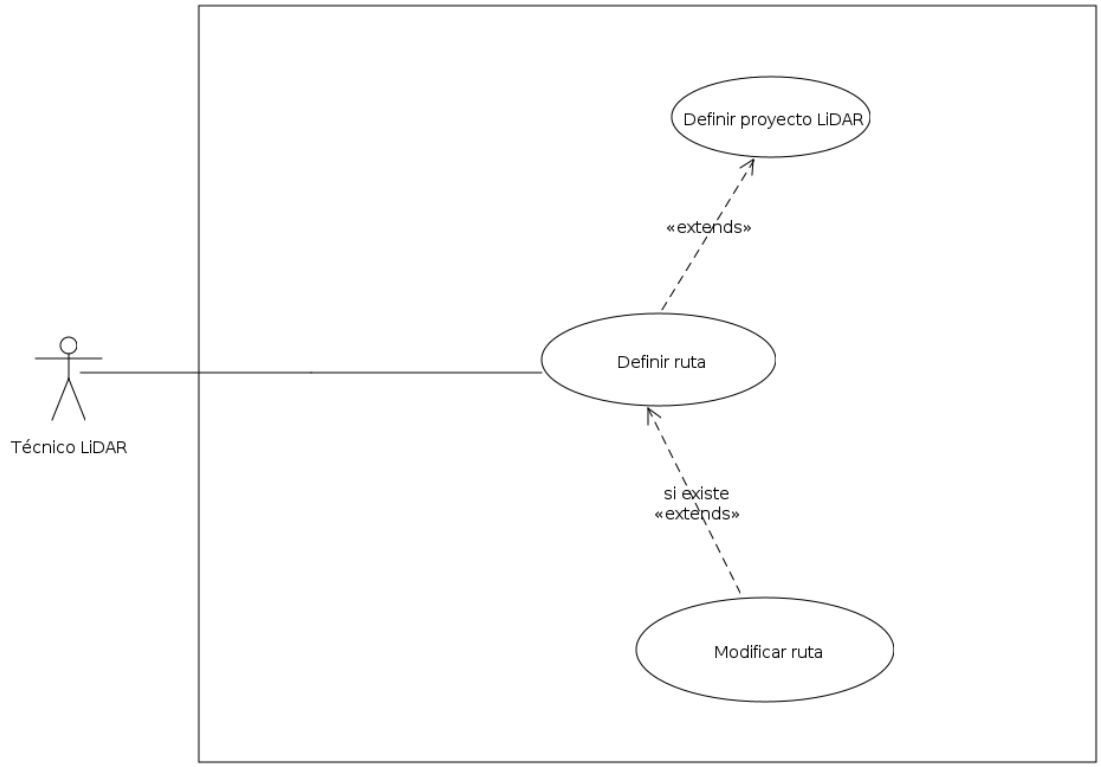


Figura 5.34: Diagrama de contexto del caso de uso definir ruta.

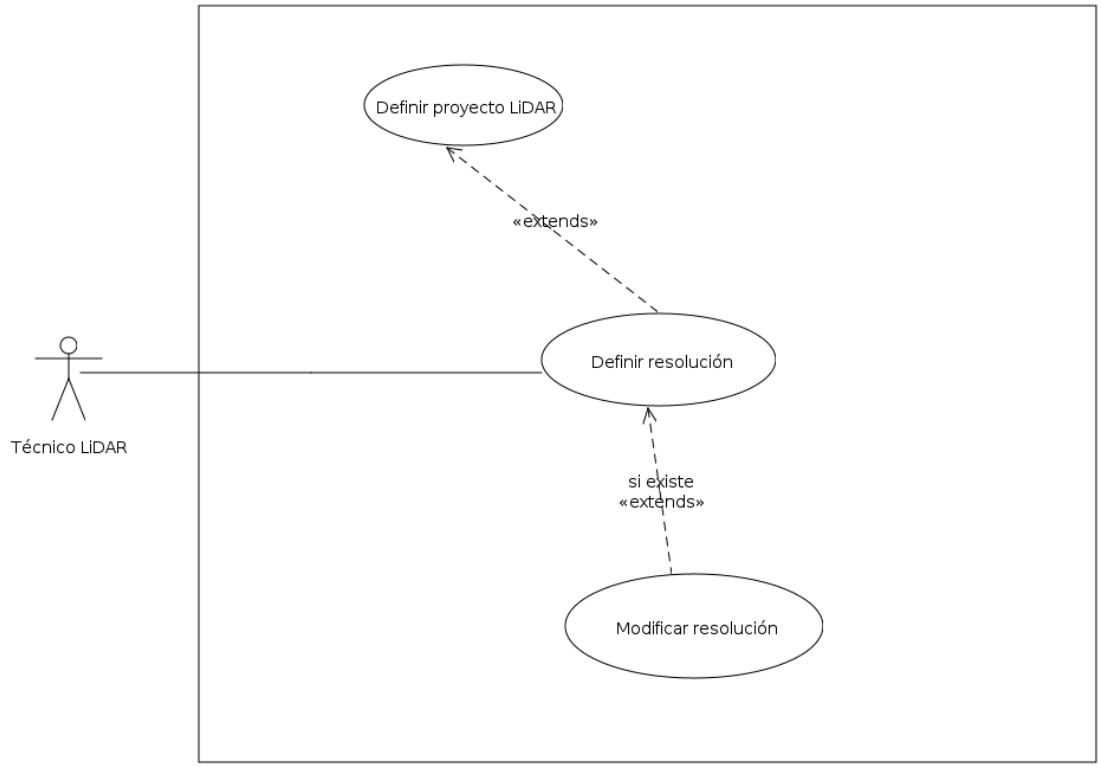


Figura 5.35: Diagrama de contexto del caso de uso definir resolución.

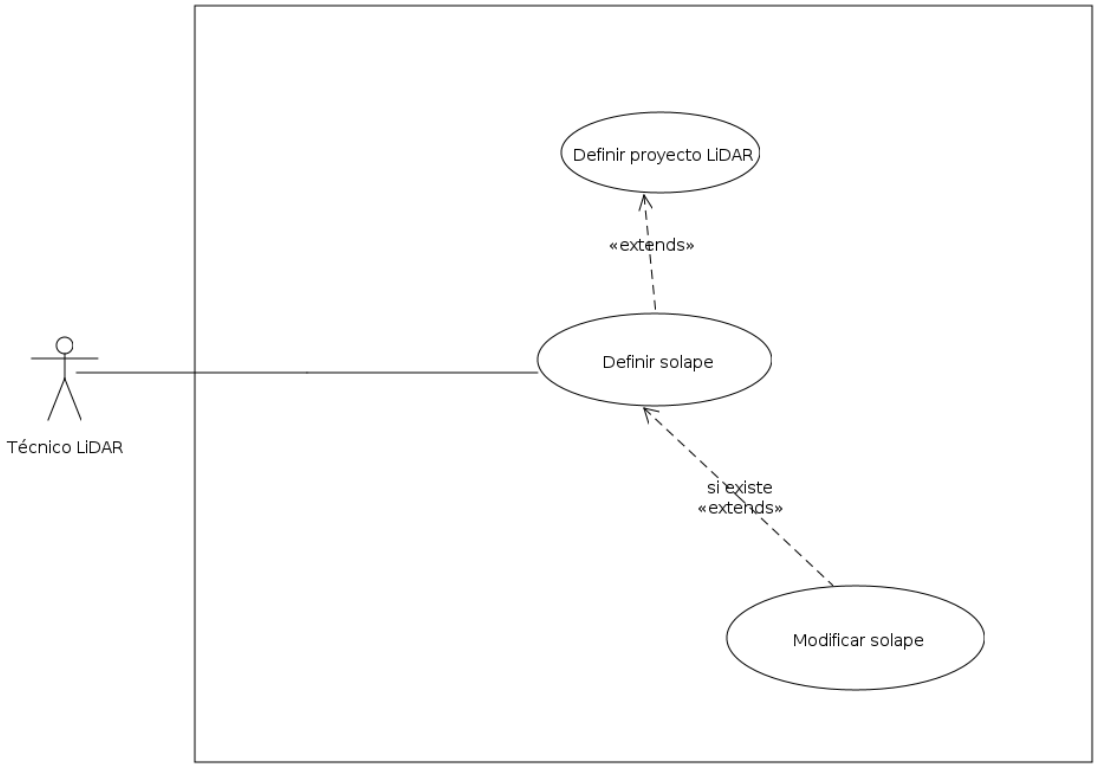


Figura 5.36: Diagrama de contexto del caso de uso definir solape.

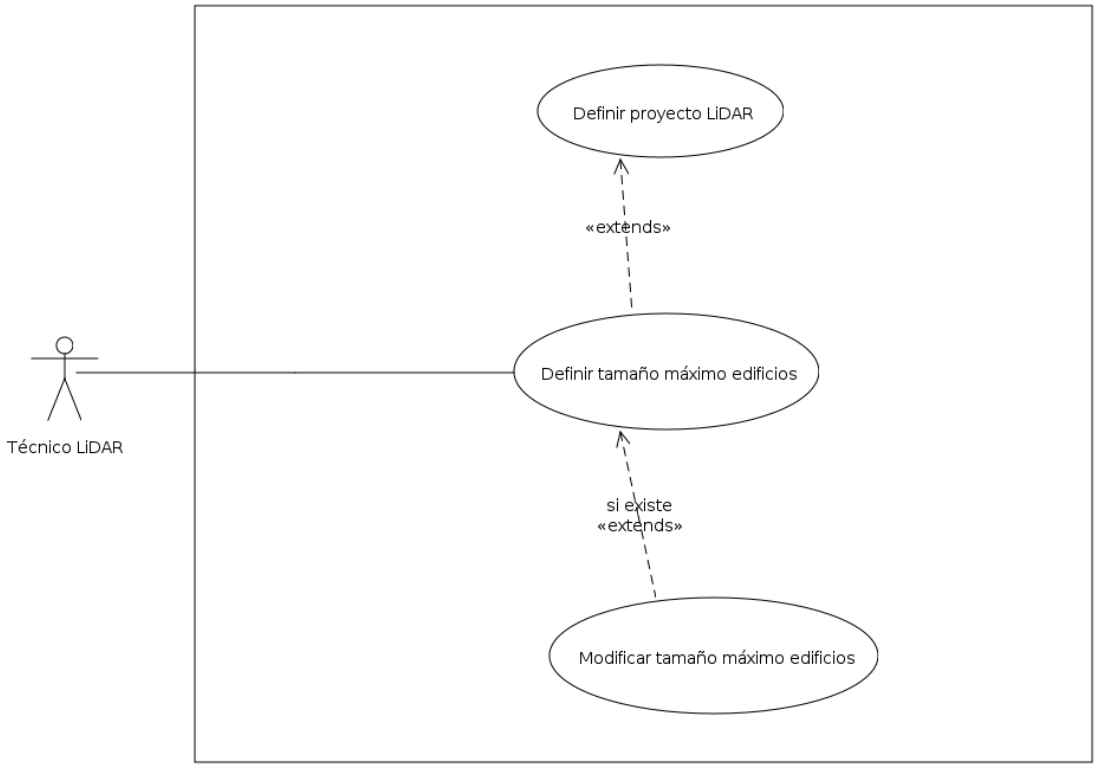


Figura 5.37: Diagrama de contexto del caso de uso definir tamaño máximo de edificio.

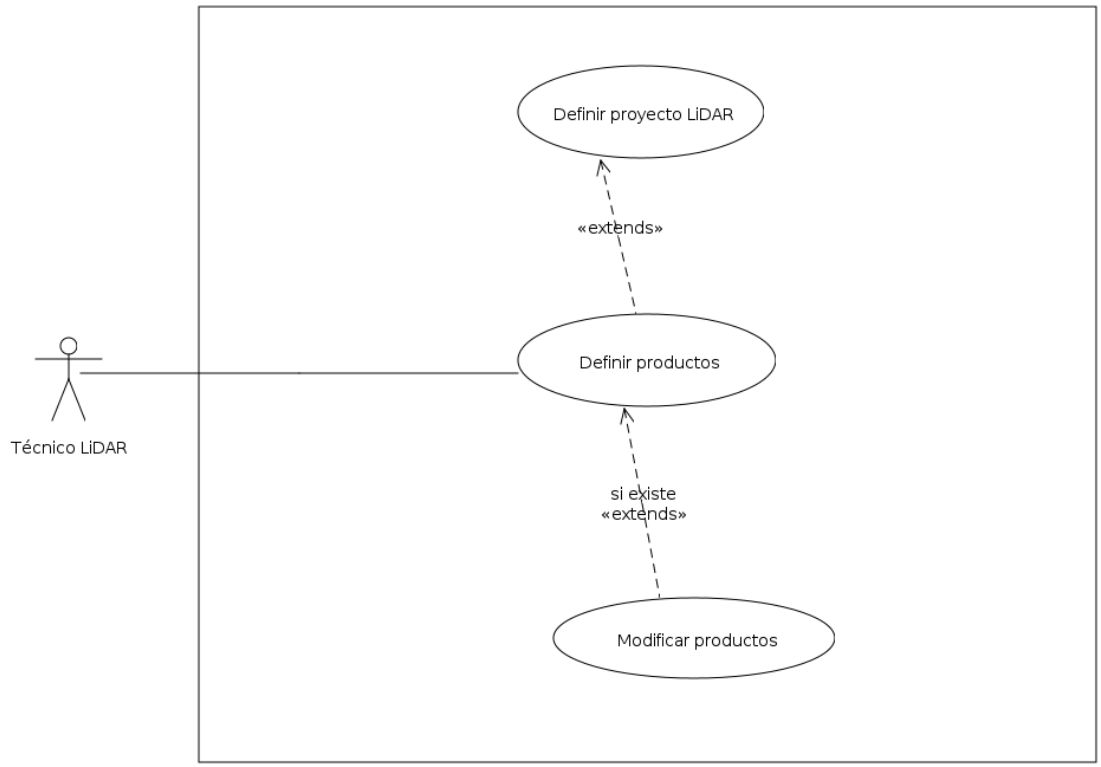


Figura 5.38: Diagrama de contexto del caso de uso definir productos.

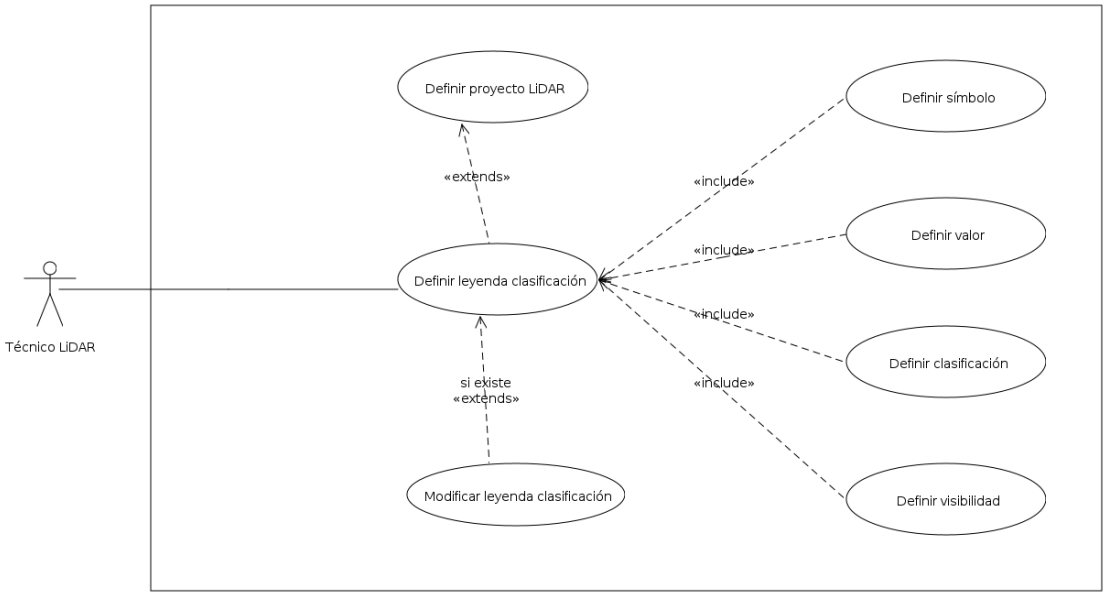


Figura 5.39: Diagrama de contexto del caso de uso definir leyenda de clasificación.

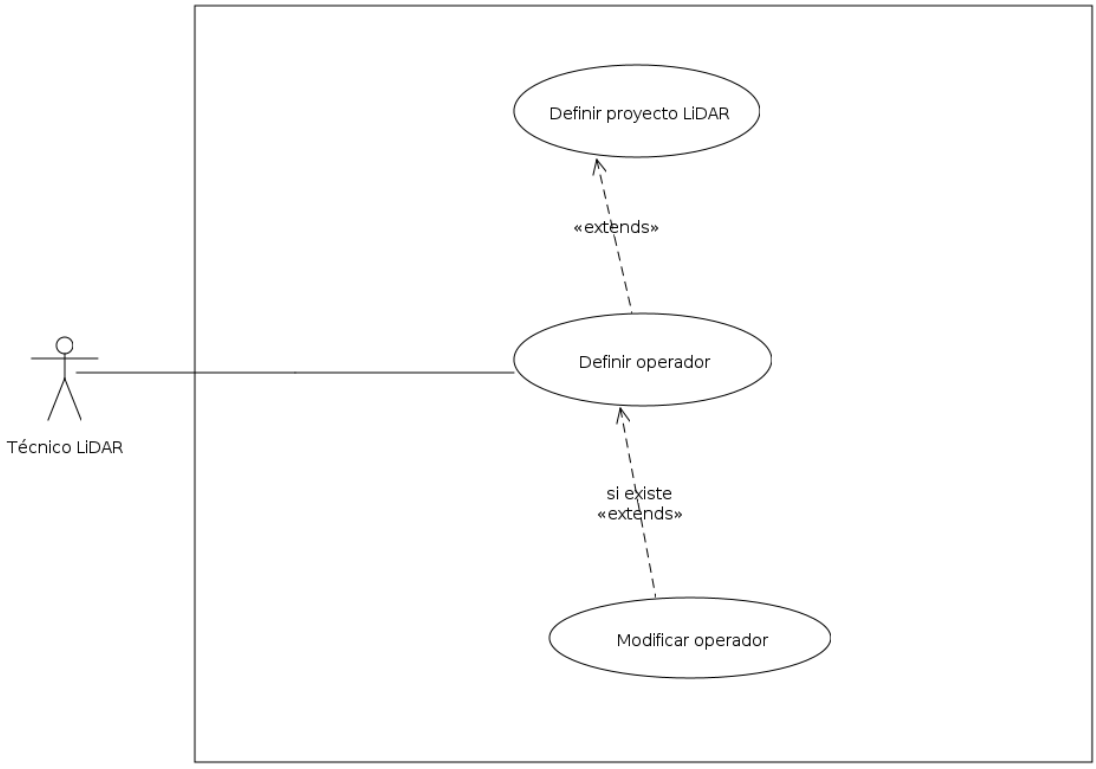


Figura 5.40: Diagrama de contexto del caso de uso definir operador.

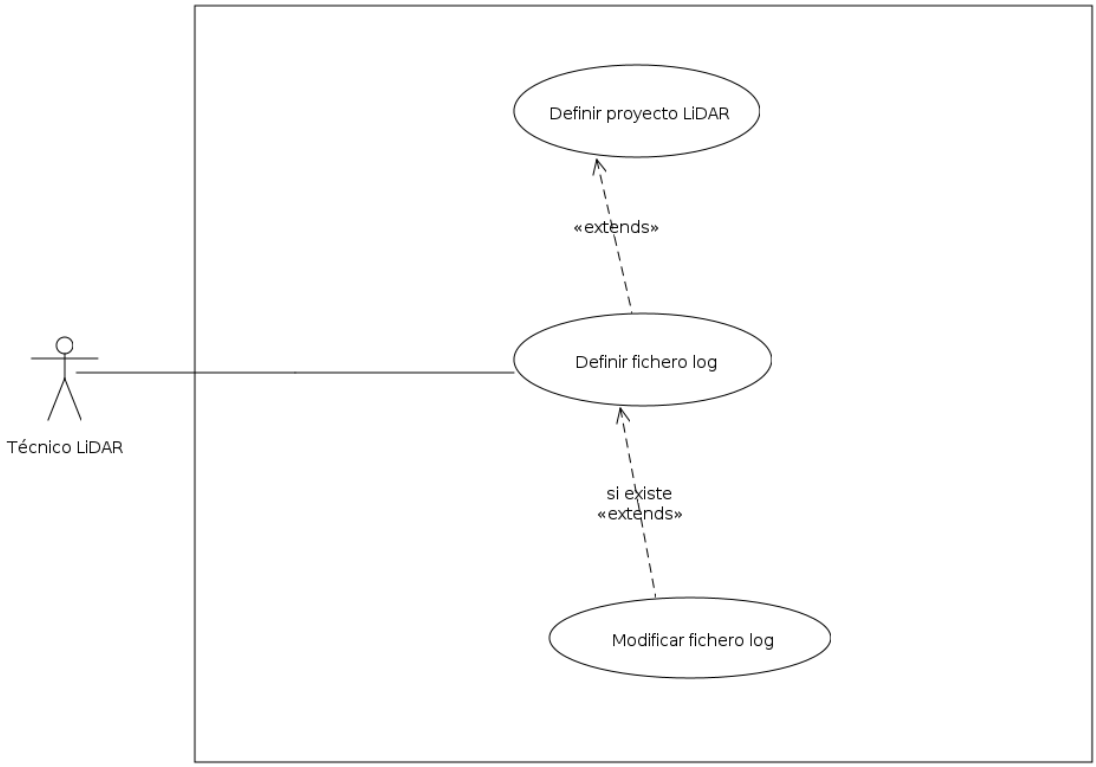


Figura 5.41: Diagrama de contexto del caso de uso definir fichero log.

CAPÍTULO 6

Diseño

6.1 Introducción

El diseño de Software juega un papel importante en el desarrollo de software ofreciendo al ingeniero de software fundamentos sobre la aplicación de métodos lo cual permite al ingeniero de software poder producir varios modelos del sistema o producto que se va a construir. y forman una especie de plan de la solución de la aplicación. Los modelos puede evaluarse en relación con su calidad y mejorarse antes de generar código, de realizar pruebas y de que los usuarios finales se vean involucrados a gran escala. El diseño establece, en gran parte, la calidad del software.

Una definición de diseño: "*El proceso de definición de la arquitectura, componentes, interfaces y otras características de un sistema o componente que resulta de este proceso*".

Una vez analizados y especificados los requisitos, el diseño del software es la última acción de la ingeniería del software correspondiente a la actividad del modelado, la cual establece una plataforma para la construcción, es decir, generación de código y prueba. Por lo tanto

una vez analizado a fondo gvSIG y la funcionalidad pretendida, es momento de abordar el diseño del proyecto.

6.2 Contexto

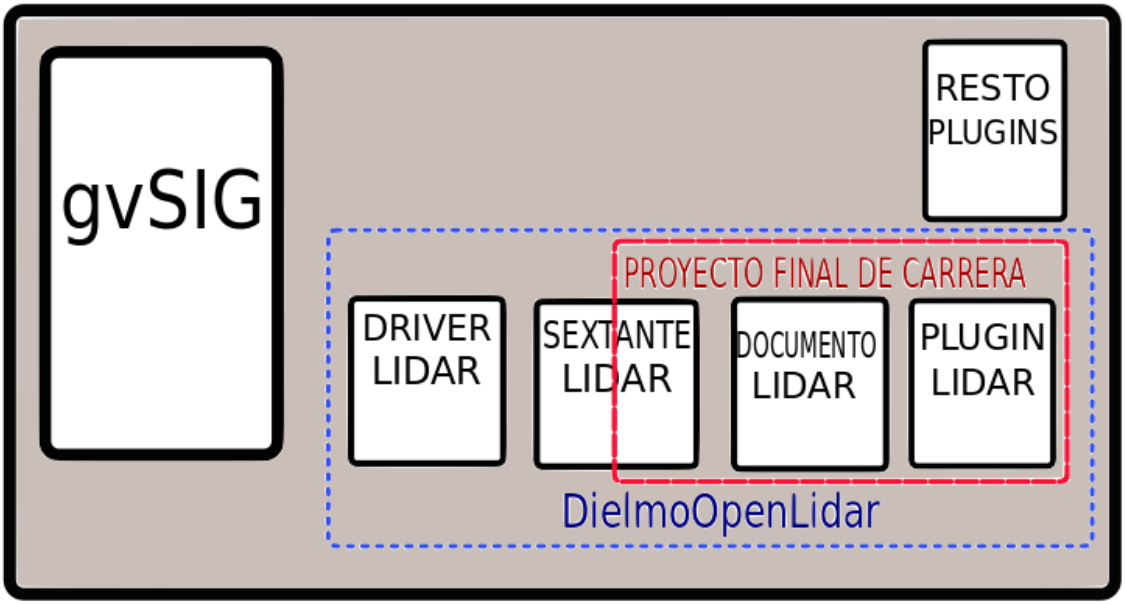


Figura 6.1: Diagrama de contexto del proyecto.

Partiendo de un Driver de lectura y escritura de datos LiDAR ya implementado por Dielmo, conociendo internamente gvSIG y teniendo marcadas las funcionalidades a obtener ya se pueden tomar decisiones en cuanto a la arquitectura de este desarrollo. Por un lado se pretende controlar gran cantidad de datos de entrada para el manejo de un proyecto. Muchos de estos datos son cartográficos, por tanto, se pretende una cómoda administración de estos datos, permitiendo persistirlos y recuperarlos en cualquier otro momento. Por otro lado se quiere poder trabajar con los componentes de gvSIG como vistas, y sus herramientas, tablas, leyendas, elementos de la interfaz de usuario como botones, menus, etc.

Por lo tanto la solución que se propone consiste en el diseño e implementación de un nuevo tipo de documento en gvSIG, que aparecerá en el gestor de proyectos junto a los ya existentes vista, tabla y mapa. Se ha diseñado y modelado toda la funcionalidad necesaria para poder persistir todos los datos de entrada necesarios para un proyecto de

datos LiDAR convencional.

Por otro lado se ha decidido diseñar e implementar un plugin que aporte una serie de extensiones que complementen la funcionalidad de gvSIG aportándole nueva funcionalidad para el trabajo con los datos definidos en el documento LiDAR creado.

Finalmente y pese a que en un principio quedaba fuera de los límites marcados para este proyecto final de carrera, se ha desarrollado algún algoritmo de análisis de datos LiDAR sobre el sistema SEXTANTE que paralelamente a este proyecto estaba siendo adaptado para esta finalidad.

Con este desarrollo Dielmo comienza a ultimar su producto DielmoOpenLidar.

6.3 Diseño del ducumento LiDAR

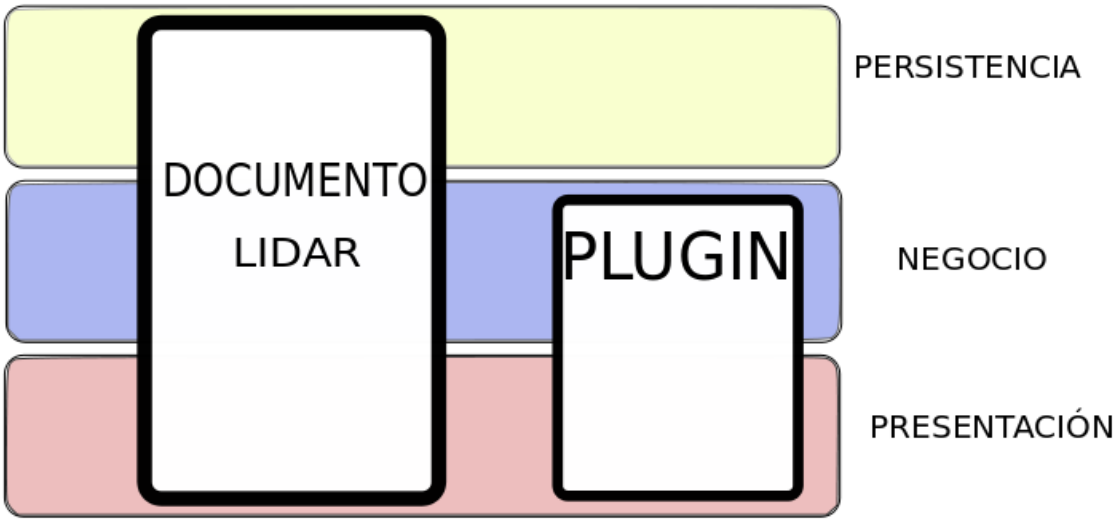


Figura 6.2: LidarDocument dentro de gvSIG.

El diseño del documento LiDAR es la parte que más profundiza en gvSIG de todo el proyecto. Abarca las tres capas: Presentación, capa media (Aplicación o de Negocio) y una capa de acceso a datos (o capa de persistencia).

La capa de presentacion consistirá en la recogida de datos mediante

paneles. Los datos son de muy diversa naturaleza con necesidades muy distintas para cada tipo de dato. Se van a recoger capas cartográficas de diferentes tipos, leyendas, valores numéricos, rutas a directorios, y demás atributos. El análisis de los elementos de gvSIG para el diseño de interfaz gráfica ha resultado esencial para la obtención de paneles versátiles para tal fin.

La capa de negocio abarca todos los procesos de cálculo y gestiones necesarias para cubrir las funcionalidades esperadas. En esta capa se calcularán las extensiones de los ficheros definidos en la capa de presentación, se abrirán capas por código, se analizarán que los datos introducidos en los paneles sean del tipo esperado, y demás funcionalidades marcadas.

La persistencia en gvSIG se realiza implementando métodos que recogen los datos a persistir en un XMLEntity y sus consiguientes métodos de lectura y reconstrucción de los objetos a partir de estas porciones de fichero xml que se agrupan en un fichero común con extensión .gvp. Por lo tanto se va a diseñar e implementar un sistema que adjunte al resto de datos persistidos al salvar un proyecto en gvSIG, los datos de entrada de un proyecto gvSIG y algunos parámetros calculados en la capa de negocio sean persistidos como si siempre hubieran formado parte de gvSIG.

En la imagen 6.14 se muestra dónde se ubica el documeto que se ha implementado en el contexto de gvSIG.

En la figura 6.4 se muestra el diagrama de secuencia que de forma esquemática muestra las interacciones entre el técnico LiDAR, el documento LiDAR, el plugin, y gvSIG, necesarias para crear un nuevo documento LiDAR.

6.4 Diseño del plugin

Las extensiones se situan más ecuadradas entre la capa de negocio y la de presentación de la aplicación. Éstas contienen cantidad de paneles de interacción con el usuario. Por el contrario tienen, en este caso,

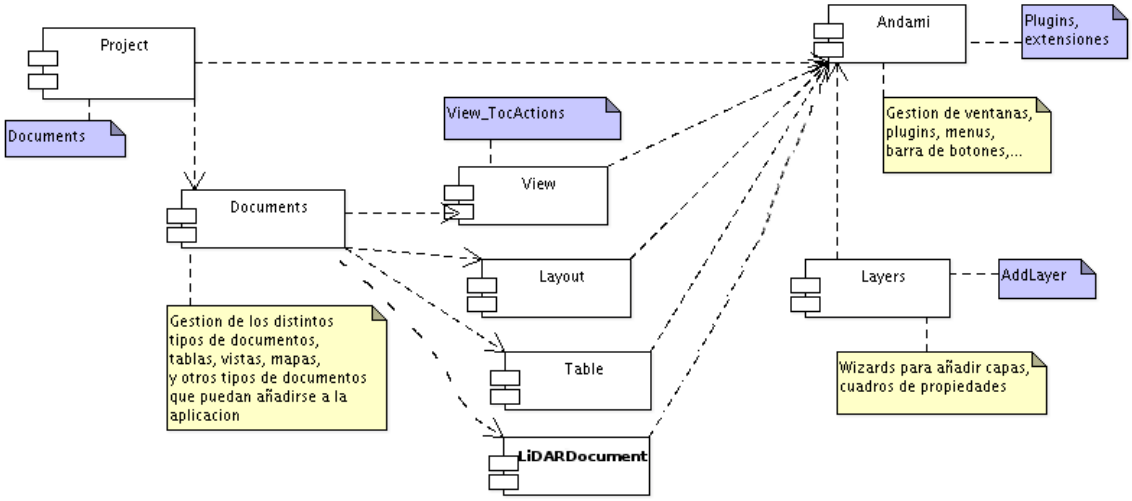


Figura 6.3: LidarDocument dentro de gvSIG.

poca presencia en la capa de persistencia. En el caso de las herramientas de conversión sí salvan ficheros, pero la tónica general es la de situarse en la capa intermedia realizando tareas y cálculos para ser presentados en paneles al usuario.
El sigiente diagrama de componentes sitúa LidarDocument dentro de gvSIG.

En cuanto al diseño de extensiones en gvSIG hay que extender de la clase abstracta Extension que implementa la interfaz IExtension. En el siguiente diagrama de clases se representa la creación de una nueva extensión en gvSIG.

Las extensiones contienen al menos una clase que extiende de la clase Extension. Esto obliga a implementar los métodos de la interfaz IExtension.

Se ha optado por crear una extensión para cada una de las funcionalidades definidas en la fase de análisis. Una extensión para el cargado automático de capas definidas, otra extensión para la visualización de perfiles longitudinales, otra para conversión de ficheros de .LAS a .XYZ, otra para el paso inverso, y finalmente la última para seleccionar el proyecto LiDAR activo. Esta última extensión no fué detectada su necesidad hasta la fase de implementación. En esta fase puso de manifiesto la necesidad de establecer el proyecto LiDAR del que se

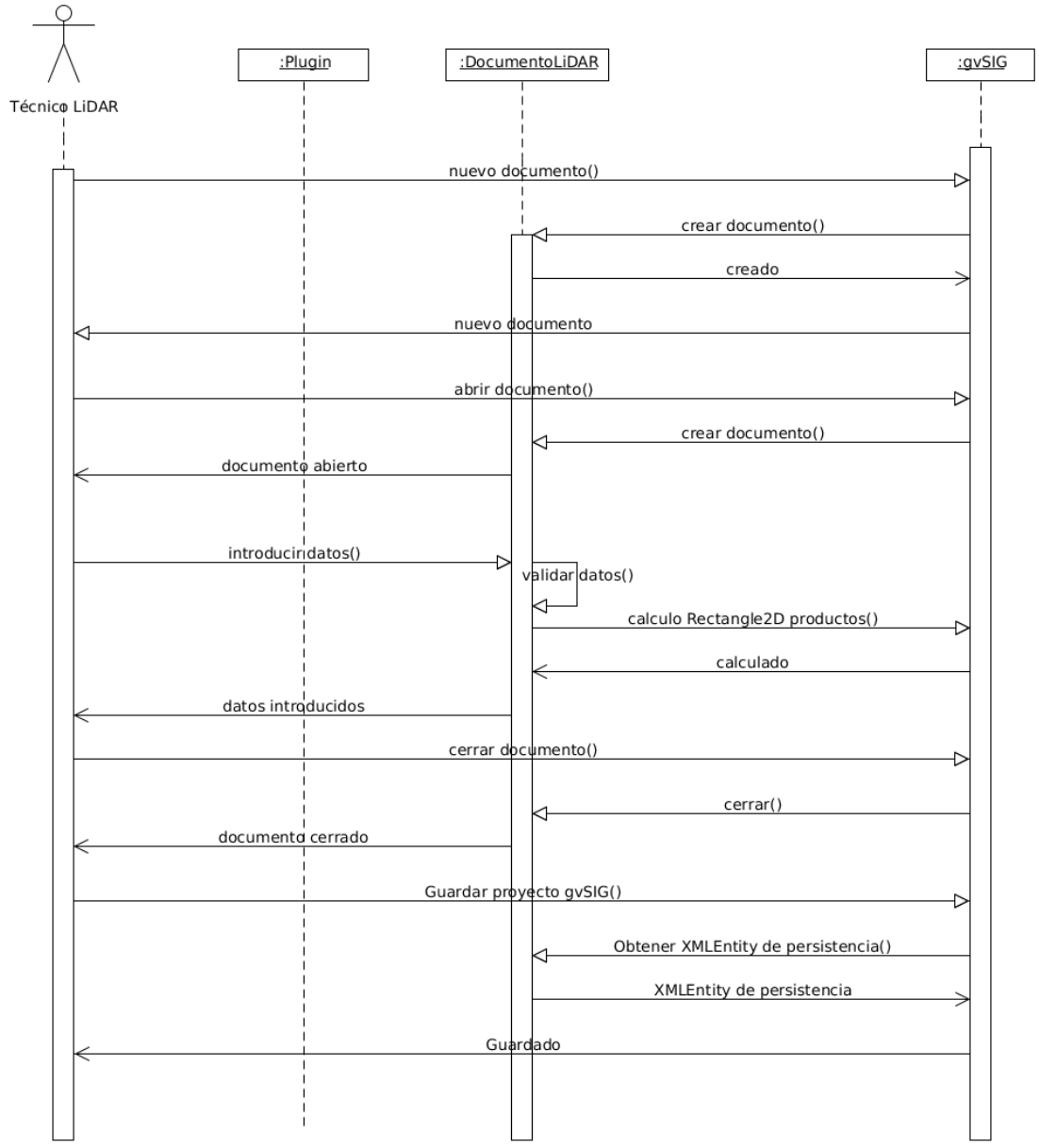


Figura 6.4: Diagrama de secuencia creación de un documento LiDAR.

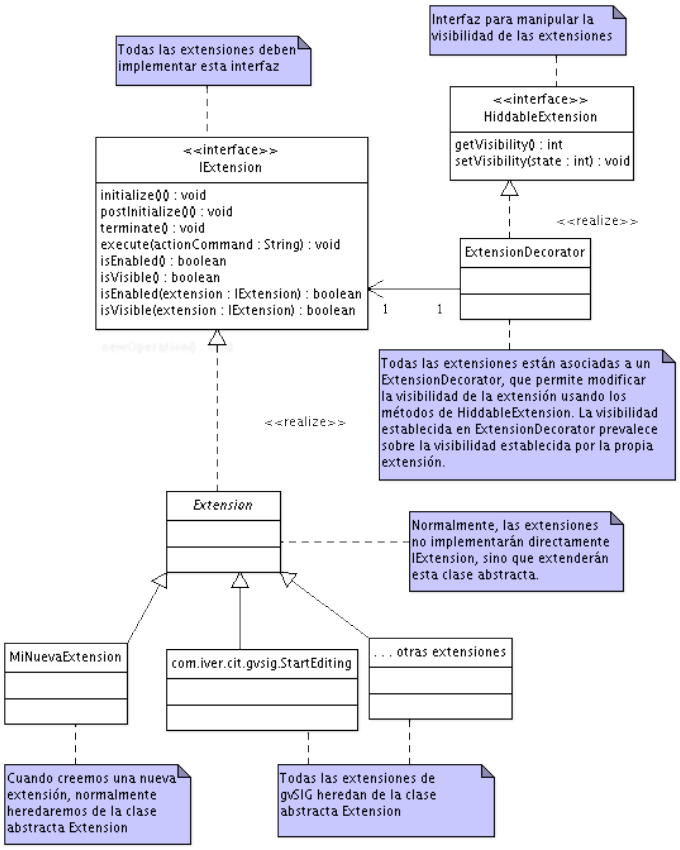


Figura 6.5: Diagrama de clases de una nueva extensión en gvSIG.

obtendrían los datos en las extensiones ya implementadas. Al poder convivir en el gestor de proyectos n documentos LiDAR simultáneamente, en extensiones como la de carga de cartografía se necesitaba saber cuál de todos representaba la fuende de datos para la extensión. En otros documentos, como en vistas o tablas esto no es necesario, pues las herramientas o extensiones que necesitan interactuar con ellas la obtienen si la ventana activa es precisamente la propia vista. Sin embargo en este caso la ventana del documento LiDAR únicamente recoge datos, produciéndose la interacción de la herramienta con una vista, y no con interfaces del documento, no pudiendo determinar con el método de la ventana activa cuál sería el documento LiDAR sobre el que se trabajaría.

```
public void initialize()
public void execute(String actionCommand)
public boolean isEnabled()
public boolean isVisible()
```

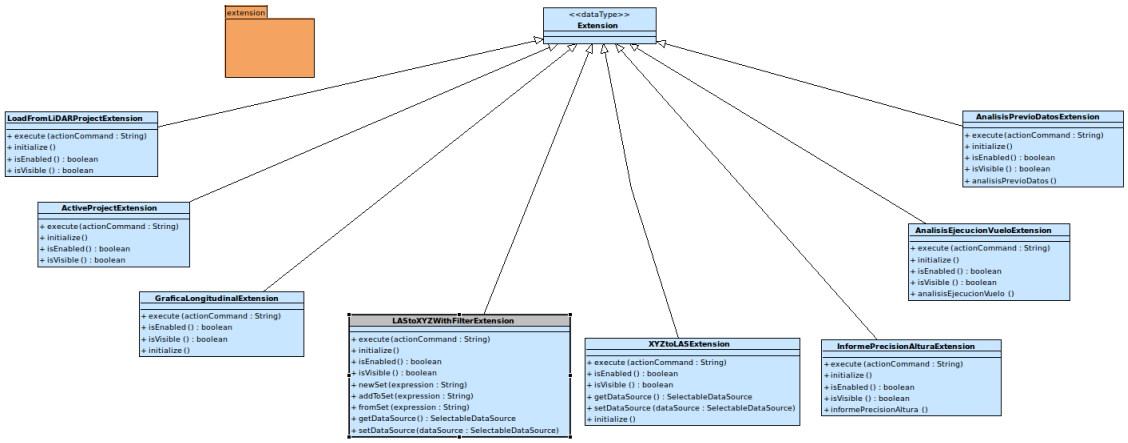


Figura 6.6: Diagrama de extensiones del plugin.

6.4.1 Diagramas de secuencia

A continuación se muestran los diagramas de secuencia de las principales funcionalidades del plugin. De forma esquemática se han representado el técnico LiDAR, el documento LiDAR, el plugin (que engloba todas las extensiones) y gvSIG (que engloba toda la funcionalidad del resto de la aplicación).

6.5 Diseño interfaces

Tanto en el documento como en el plugin se hace uso de elementos que aparecen en la capa de presentación en forma de paneles de recogida de datos. El documento recoge gran cantidad de parámetros de entrada para el proyecto. Las extensiones también presentan nuevos paneles para en algunos casos mostrar resultados, como es el caso del visualizador de gráficas longitudinales, y para la recogida de datos como es el caso de la selección del proyecto lidar activo. Los paneles que se han diseñado tienen este aspecto:

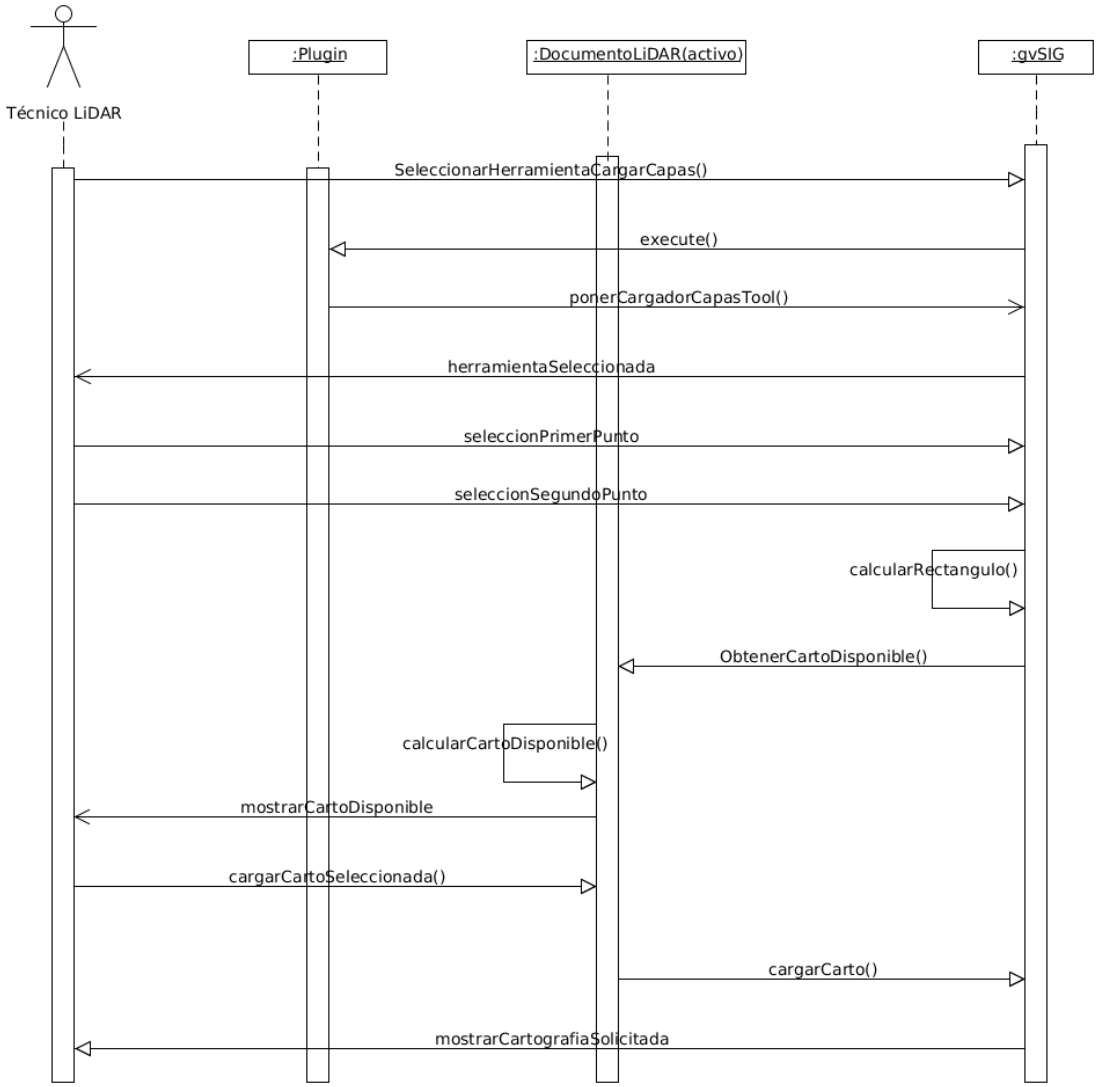


Figura 6.7: Diagrama de secuencia cargador de capas avanzado.

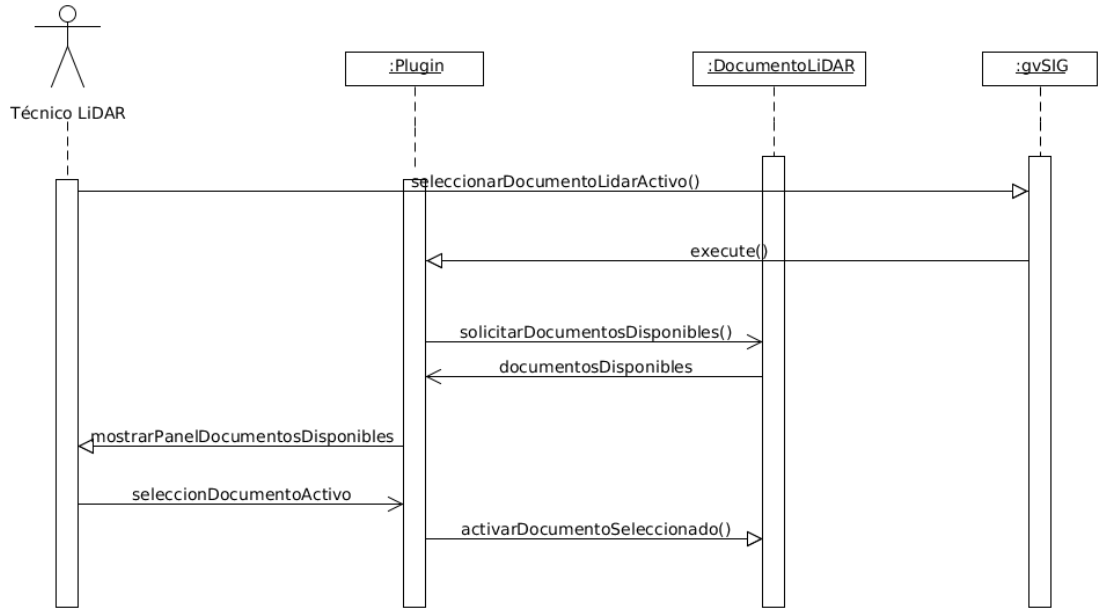


Figura 6.8: Diagrama de secuencia selección del documento LiDAR activo.

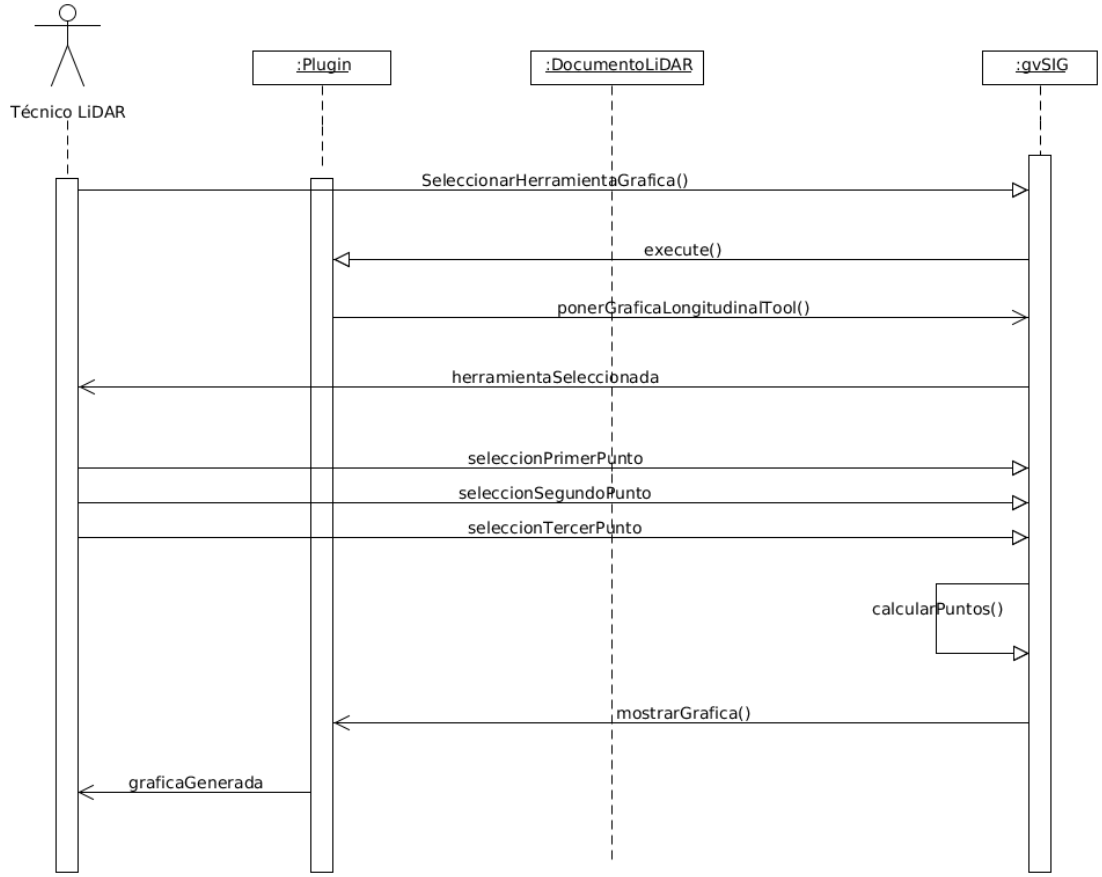


Figura 6.9: Diagrama de secuencia visualización de gráficas longitudinales.

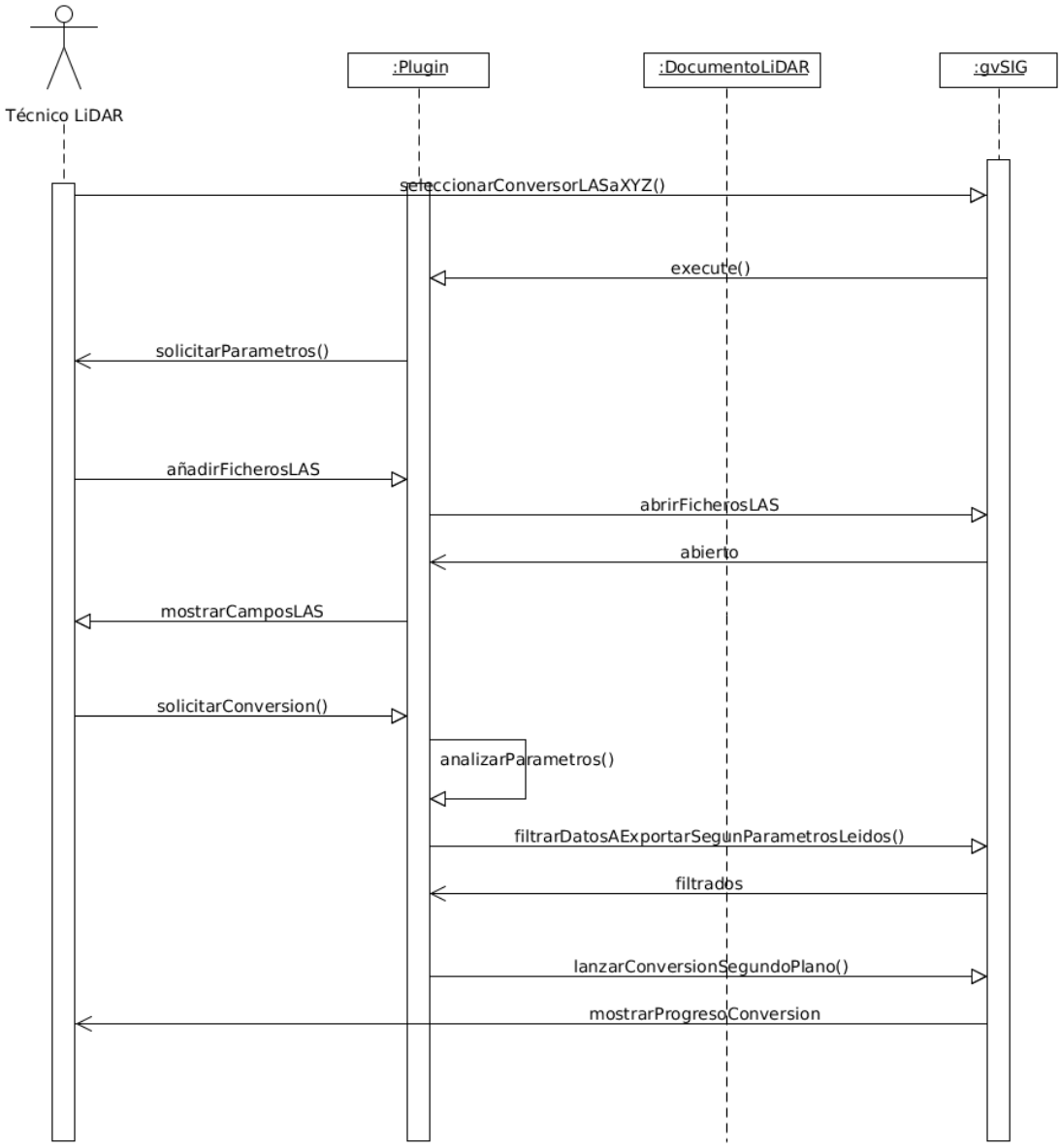


Figura 6.10: Diagrama de secuencia conversor avanzado de .LAS a .XYZ.

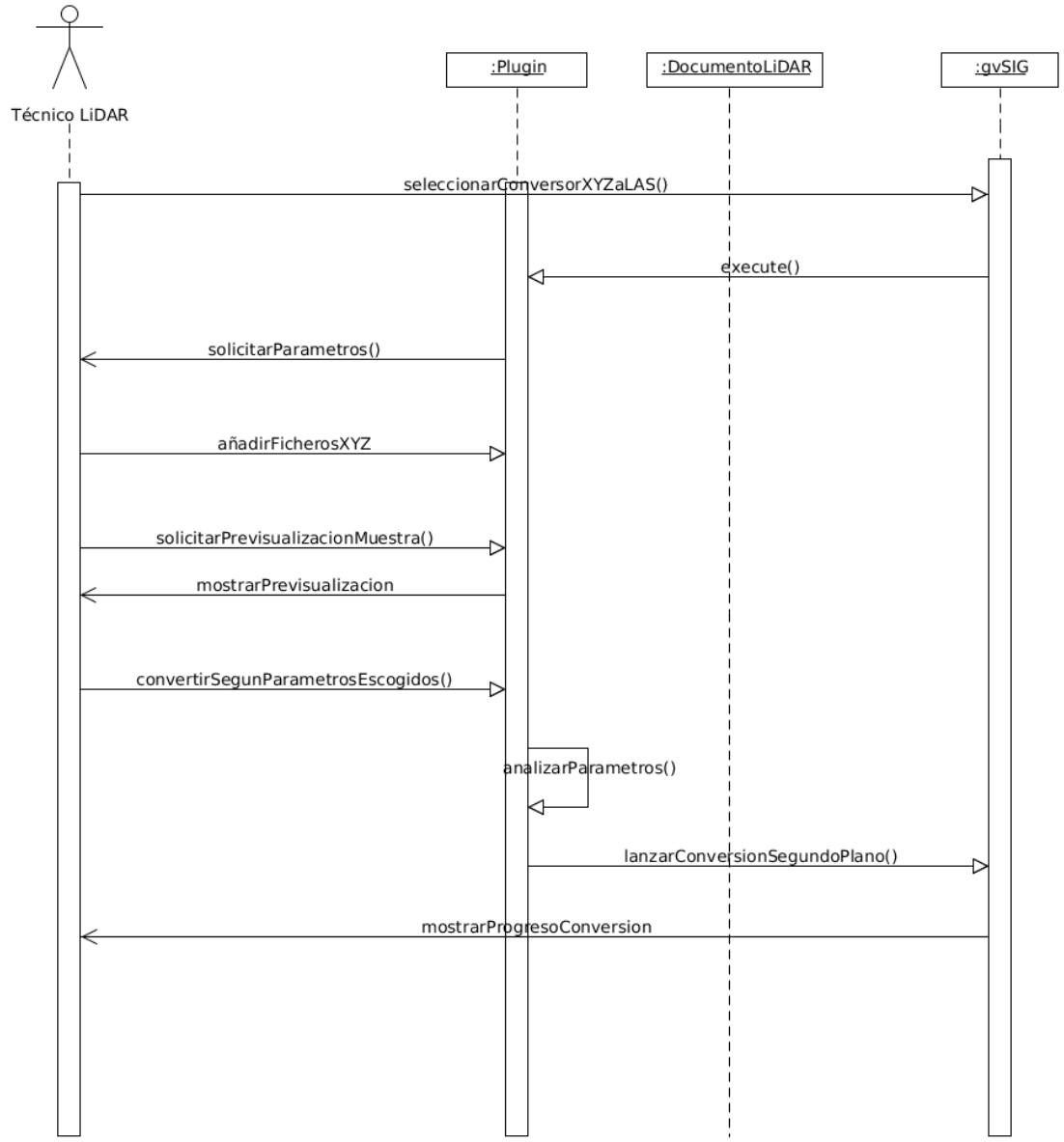


Figura 6.11: Diagrama de secuencia conversor de .XYZ a .LAS.

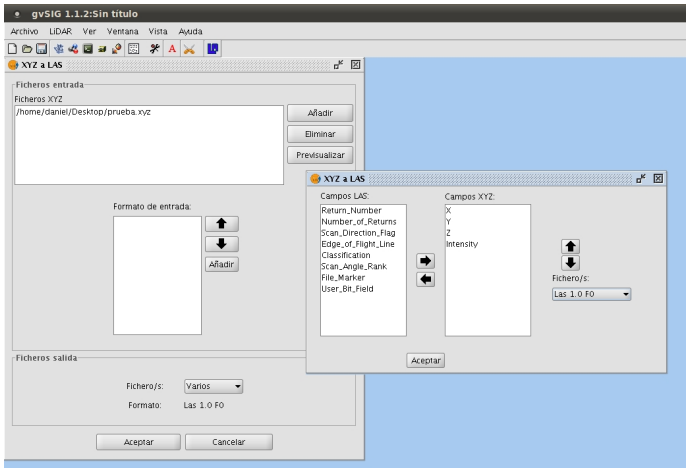


Figura 6.12: Interfaz herramienta de conversión de .xyz a .las

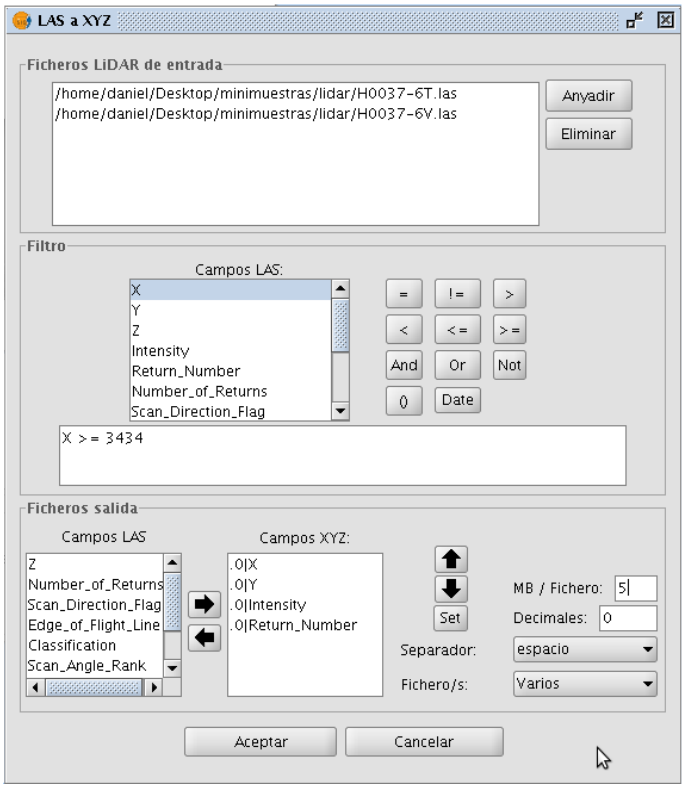


Figura 6.13: Interfaz herramienta de conversión de .las a .xyz

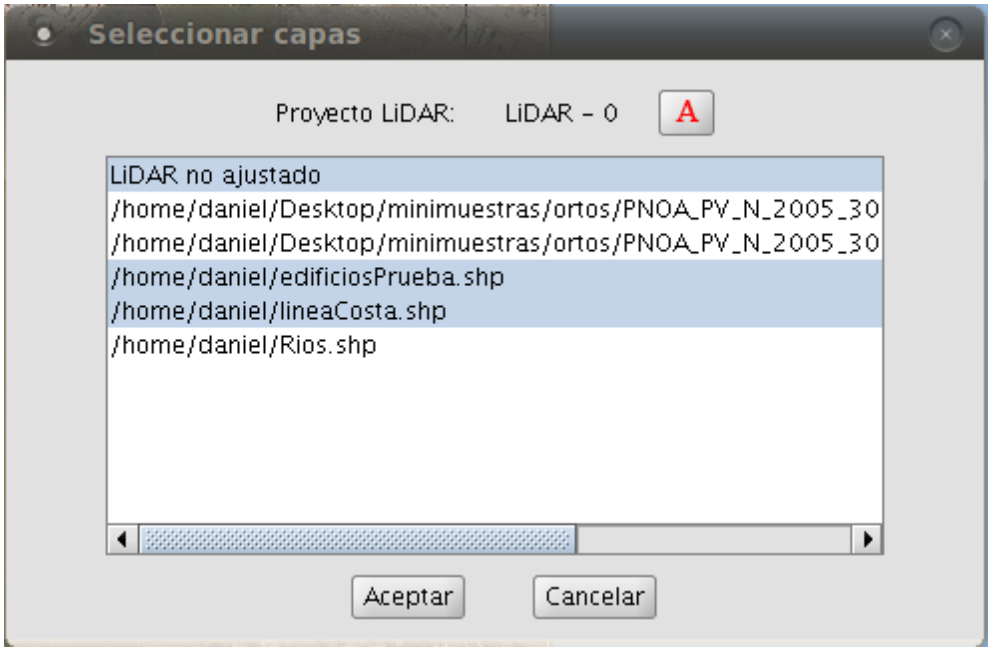


Figura 6.14: Interfaz herramienta de carga avanzada de capas.

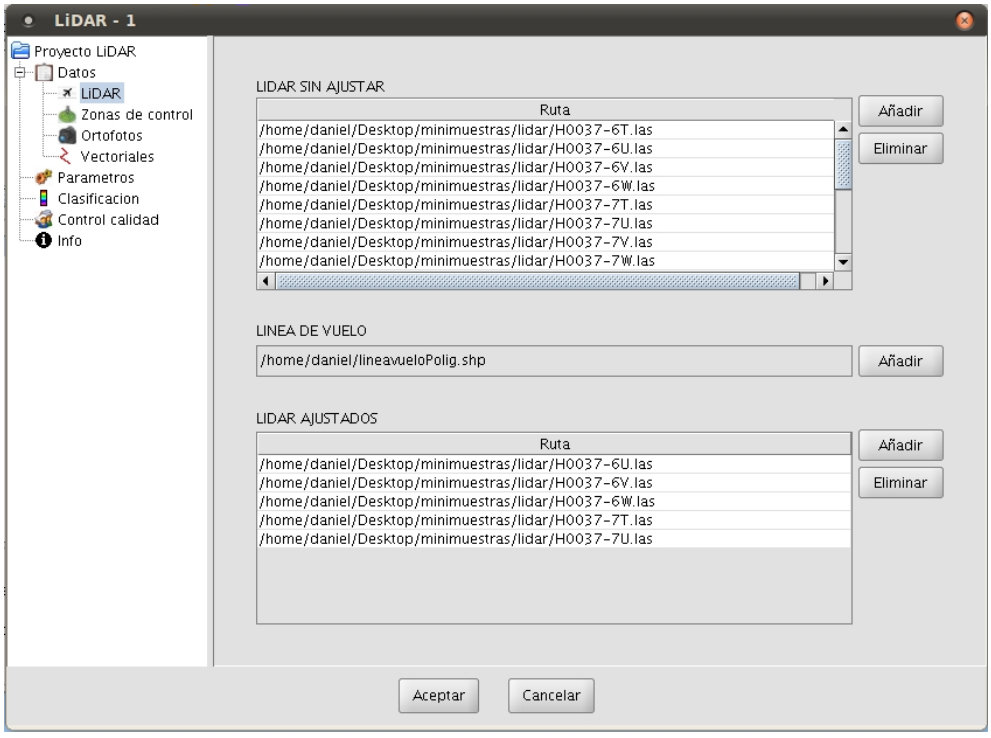


Figura 6.15: Interfaz recogida de datos del documento LiDAR.

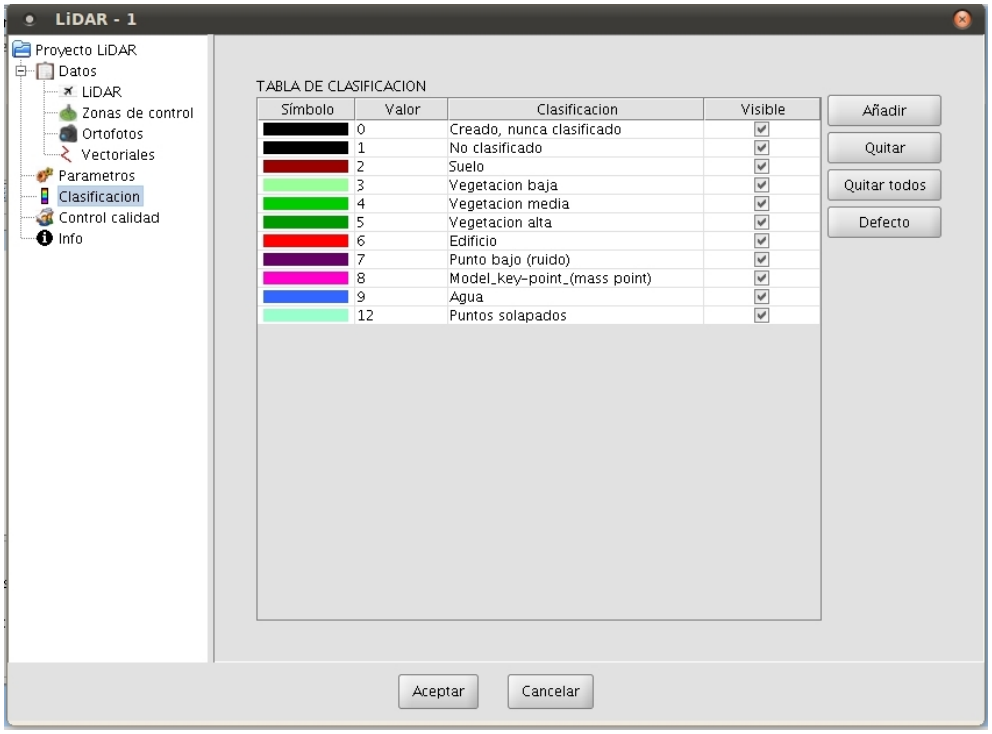


Figura 6.16: Interfaz para definir leyenda en el documento LiDAR.

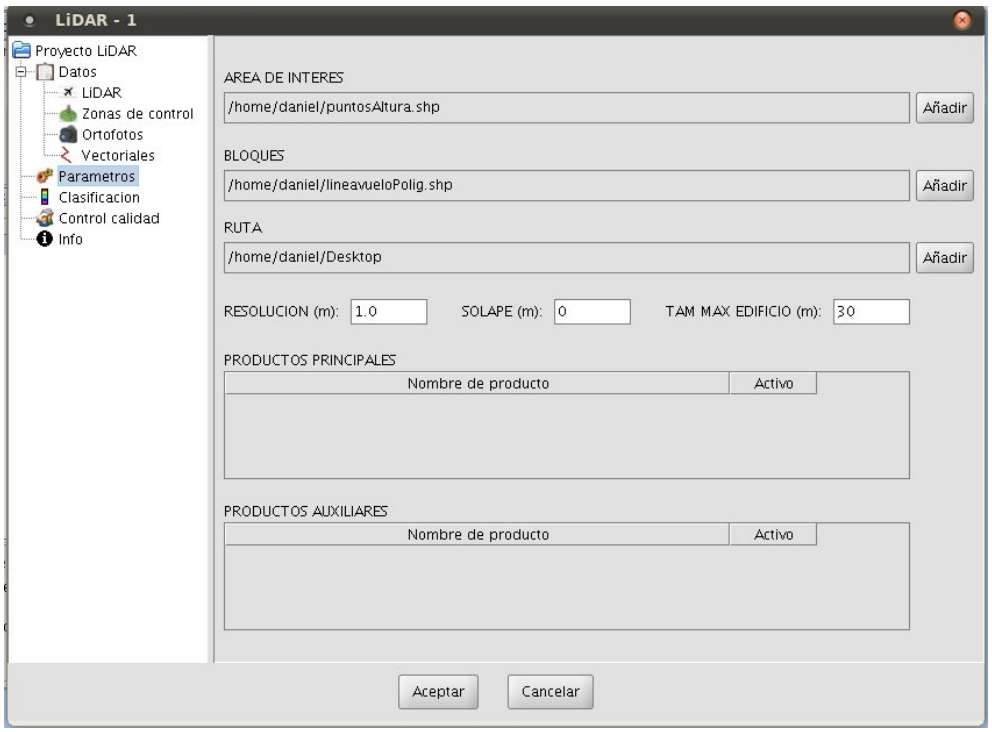


Figura 6.17: Interfaz panel recogida de parámetros del documento LiDAR.

CAPÍTULO 7

Implementación

7.1 Introducción

En este capítulo se va a detallar el proceso de implementación de la funcionalidad descrita anteriormente. Para ello se va a analizar, en primer lugar,el entorno utilizado, en este caso eclipse, seguidamente la implementación del documento LiDAR, a continuación el plugin de gvSIG desarrollado y sus extensiones, y finalmente los algoritmos de análisis.

7.2 Eclipse

Eclipse es una plataforma pensada para el desarrollo de aplicaiones. La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal, un entorno de desarrollo integrago abierto y extensible para todo y nada en particular". Se trata de una plataforma de código abierto y provee al programador de frameworks para el desarrollo de aplicaciones gráficas, modelos de software, diseño web, etc.

Figura 7.1: Entorno Eclipse.

El SDK de Eclipse incorpora herramientas de desarrollo Java y ofrece un compilador. La depuración del código mediante puntos de ruptura, para ver el estado de los objetos y variables resulta tarea fácil en Eclipse. Permite técnicas avanzadas de análisis de código. Es posible incorporar un plugin para conexión a un sepositorio de código svn, muy útil en proyectos colaborativos como es el caso. Se utilizó una conexión al svn de gvSIG en el proceso de descarga del código fuente de la aplicación.

La utilización de Eclipse ha venido marcada por gvSIG. En gvSIG aportan documentación para configurar el arranque de la aplicación y su comunidad de desarrolladores utiliza esta herramienta para desarrollo de plugins, extensiones y mejoras de la aplicación.

Diseñar un plugin con extensiones en gvSIG supone la creación en Eclipse de un proyecto. Siguiendo la filosofía impuesta, este proyecto debe comenzar por ext. En nuestro caso el proyecto se llama extProyectoLidar. Éste proyecto se ha diseñado para contener tanto la implementación del documento LiDAR como la del plugin que contiene las extensiones.

7.3 Implementación del documento LiDAR

La implementación de un nuevo tipo de documento en gvSIG supone un reto de valor añadido frente al desarrollo más habitual de plugins y extensiones. Un documento en gvSIG se presenta en el gestor de proyectos (ver figura 7.3), donde se crea, se manipula y se elimina. Además tiene la particularidad que debe ser persistido. Al salvar el proyecto en gvSIG debe almacenarse el documento en su fichero de persistencia *.gvp*. Este fichero es un xml donde se guarda toda la información del proyecto, estado de las vistas, tablas, mapas y demás documentos existentes, como es el caso del documento LiDAR que se ha desarrollado.

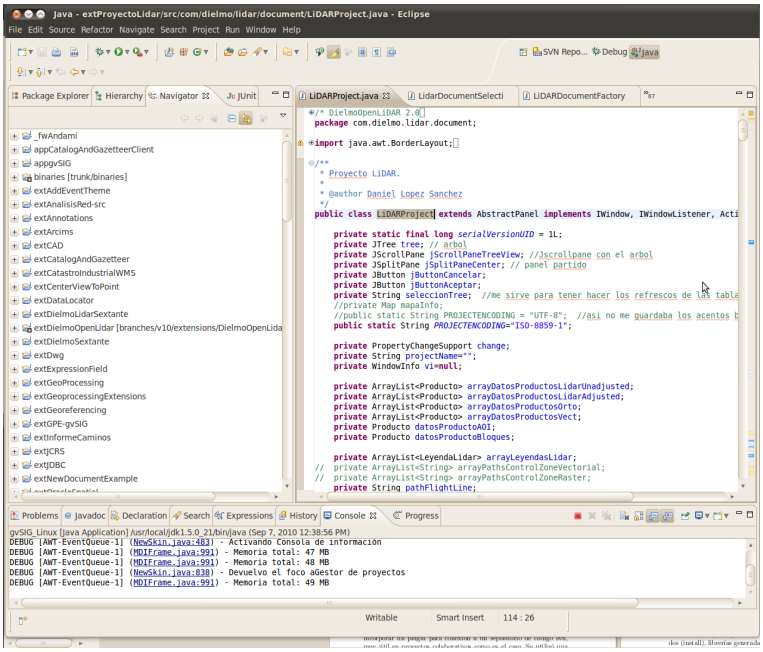


Figura 7.2: Logo de Eclipse.

Figura 7.3: Documentos en gvSIG junto al documento LiDAR desarrollado.

Para el desarrollo del documento se han implementado las clases LiDARDocoment y LiDARDocumentFactory que extienden de ProjectDocument y ProjectDocumentFactory respectivamente. La primera es la clase de la que heredan el resto de documentos. La segunda es la encargada de crear los documentos, y gestionar los posibles documentos que en el futuro pueden ampliar este modelo de objetos, como es el caso de este documento LiDAR.

El package com.iver.cig.gvsig.project es un buen punto de entrada para adentrarse en gvSIG. En teoría, sigue una organización influída por la organización por documentos dentro de un proyecto. Las clases más importantes ahí con Project.java y ProjectFactory.java. La primera representa el proyecto de gvSIG, y el método estático createFromXML() es el punto de entrada para ver cómo se lee un archivo

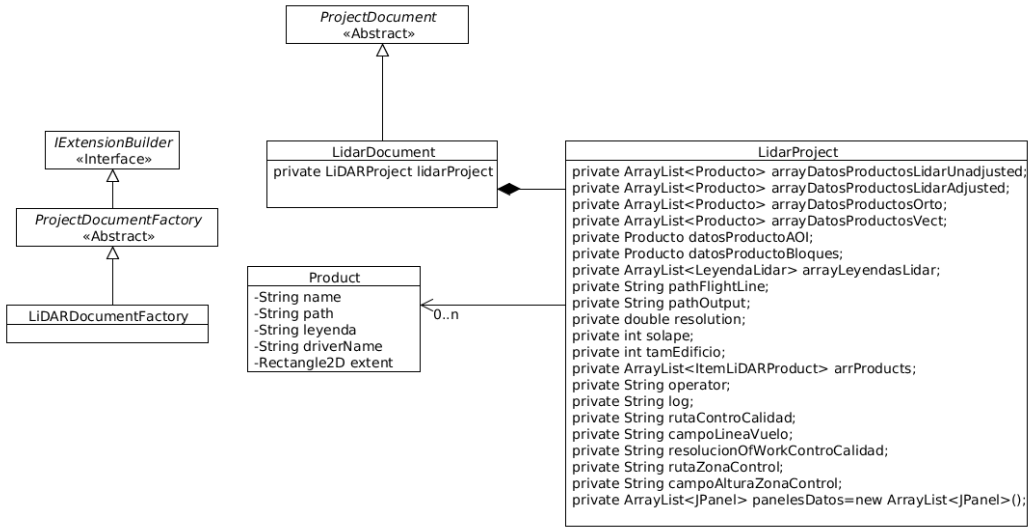


Figura 7.4: Diagrama de clases simplificado de LidarDocument.

.gvp que representa un proyecto de gvSIG. El paso contrario, crear un fichero .gvp se hace en el método writeProject de la extensión ProjectExtension, y llama al método getXMLEntity() de la clase Project.

7.3.1 **Persistencia en gvSIG.**

Como se ha comentado en el punto anterior los documentos en gvSIG deben aportar métodos para asegurar su persistencia en el fichero *.gvp* que se genera al salvar un proyecto. A continuación se analizan algunos métodos de persistencia.

El siguiente método de LiDARDocument proporciona un objeto tipo XMLEntity que contiene la información a persistir del documento LiDAR. El XMLEntity se obtiene llamado a la instancia lidarProject de LiDARDocument.

```
public XMLEntity getXMLEntity() throws SaveException {
       XMLEntity xml= super.getXMLEntity();
       try {
           xml.addChild(lidarProject.getXMLEntity());
       } catch (XMLException e) {
           // TODO Auto-generated catch block
           e.printStackTrace();
       }
       return xml;
   }
```

A continuación se muestra una pequeña porción del método de Li-

DARProject que es el que en última instancia crea el XMLEntity y coloca en él los datos:

```
public XMLEntity getXMLEntity() throws XMLException {

       XMLEntity xml = new XMLEntity();
       xml.putProperty("className", this.getClass().getName());
       xml.putProperty("resolution", resolution);
       xml.putProperty("solape", solape);
       xml.putProperty("tamEdificio", tamEdificio);
       xml.putProperty("pathOutput", pathOutput);
       xml.putProperty("pathFlightLine", pathFlightLine);
       xml.putProperty("operator", operator);
       xml.putProperty("log", log);

       ...
```

Finalmente se muestra una porción del gvp tras guardar un documento LiDAR:

```
  <xml-tag>
       <property key="className" value="LiDAR_Proyect"/>
       <property key="comment"/>
       <property key="creationDate" value="6/09/10 22:35"/>
       <property key="name" value="LiDAR - 1"/>
       <property key="owner"/>
       <xml-tag>
           <property key="className" value="com.dielmo.lidar.document.LiDARProject"/>
           <property key="resolution" value="1.0"/>
           <property key="solape" value="0"/>
           <property key="tamEdificio" value="30"/>
           <property key="pathOutput" value=""/>
           <property key="pathFlightLine" value=""/>
           <property key="operator" value=""/>
           <property key="log" value=""/>

...
```

Otro trozo de código en el que se guardan las ortofotos definidas y cierta información extra para la carga automática de capas, como es el caso del extent y el driver de lectura en gvSIG:

```
       for(int i=0;i<arrayDatosProductosOrto.size();i++){

           XMLEntity xmlchild =new XMLEntity();
           Producto prod=arrayDatosProductosOrto.get(i);
           xmlchild.putProperty("Nombre", prod.getName());
           xmlchild.putProperty("Driver", prod.getDriverName());
           xmlchild.putProperty("Path", prod.getPath());
           xmlchild.putProperty("Extension","");
           xmlchild.putProperty("ExtentX",prod.getExtension().getX());
           xmlchild.putProperty("ExtentY",prod.getExtension().getY());
           xmlchild.putProperty("ExtentWidth",prod.getExtension().getWidth());
           xmlchild.putProperty("ExtentHeight",prod.getExtension().getHeight());

           xmlchildOrtos.addChild(xmlchild);
       }
```

Como se ha mostrado en el capítulo anterior, el documento LiDAR está compuesto por una serie de paneles que recogen toda la información definida por el técnico, usuario de gvSIG. Cada uno de los paneles está destinado a fines concretos, siendo en varios de ellos necesario introducir ficheros de un determinado tipo como entrada. Concretando en uno de ellos, por ejemplo el de recogida de datos LiDAR, al pulsar al botón añadir, el panel espera obtener como entrada únicamente ficheros LiDAR. Para esta finalidad la clase PanelLidarOption al crear el botón implementa el siguiente cógdigo:

```
JButton boton = new JButton(PluginServices.getText(null,"Anyadir"));

          boton.addActionListener(new ActionListener() {

              public void actionPerformed(ActionEvent e) {

                      JFileChooser fileChooser = new JFileChooser();
                      String [] extensiones ={"las","bin"};
                      fileChooser.addChoosableFileFilter(new GenericFileFilter("las",
                          PluginServices.getText(null,"LAS_Files")));
                      fileChooser.addChoosableFileFilter(new GenericFileFilter("bin",
                          PluginServices.getText(null,"BIN_Files")));
                      fileChooser.addChoosableFileFilter(new GenericFileFilter(extensiones
                          , PluginServices.getText(null,"BIN_Files")+" "+PluginServices.
                          getText(null,"LAS_Files")));
                      File files[];
                      fileChooser.setMultiSelectionEnabled(true);
                      int result = fileChooser.showOpenDialog(
                          getBotonAnyadirRutaLidarAdjusted());
                      boolean hayErroneos=false;
                      boolean repetidos=false;
                      if (result == JFileChooser.APPROVE_OPTION && (files = fileChooser.
                          getSelectedFiles()) != null) {
                          for(int i=0; i< files.length; i++){
                              String s =files[i].getAbsolutePath();
                              FiltroTipoArchivos filtro =new FiltroTipoArchivos();
                              if (filtro.esLidar(files[i])){
                                  boolean esta=false;
                                  for(int j=0;j<ArrayDatosLiDARAdjustedModified.size();j++)
                                      {
                                      if(ArrayDatosLiDARAdjustedModified.get(j).getPath().
                                          equals(s)){
                                          esta=true;
                                          repetidos=true;
                                      }
                                  }
                                  if(!esta){
                                      //creamos un producto
                                      Producto p = new Producto();
                                      p.setDriverName("gvSIG LIDAR driver");
                                      p.setPath(s);
                                      Rectangle2D rect=null;
                                      p.setName(PluginServices.getText(null, "LIDAR_AJUSTADO
                                          "));
                                      ArrayDatosLiDARAdjustedModified.add(p);
                                  }
```

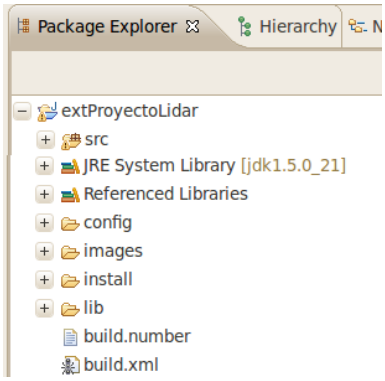


Figura 7.5: Vista del plugin en el Package Explorer de Eclipse.

```
                        }
                        else  hayErroneos=true;
                    }
                    if(hayErroneos){
                        JFrame j= new JFrame();
                        JOptionPane.showMessageDialog(j,PluginServices.getText(null,"
                            Algunos_lidar_no"));
                    }
                    if (repetidos){
                        JFrame j= new JFrame();
                        JOptionPane.showMessageDialog(j,PluginServices.getText(null,"
                            Repetidos_no"));
                    }
                    initialize();
                    refrescarInfo();
                }
            }
        });
```

Al botón se le incorpora un *JFileChooser* al que se le ha incorporado filtros para únicamente permitir ficheros con la extensión esperada. Una vez recogidos esos datos son almacenados en un arraylist

7.4 Implementación del plugin

Para el desarrollo del plugin se ha creado un nuevo proyecto en Eclipse (ver figura 7.5) donde se situa el código fuente (src), imágenes utilizadas (images), generador de instaladores e instaladores generados (install), librerías generadas (lib) y el fichero config.xml.

7.4.1 Dependencias del proyecto

Las dependencias con otros proyectos se muestran en la figura 7.6. _fwAndami, appgvSIG, libFmap, extDielmoOpenLidar, libDielmoOpenLiDAR, libDielmoLiDARAlgorithms, sextante y sextante_gui son los

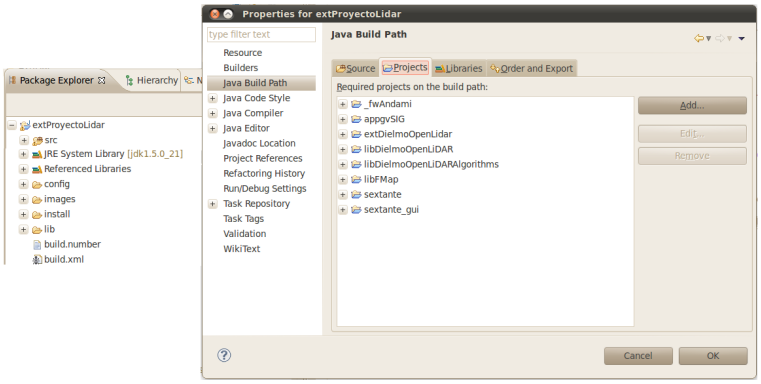


Figura 7.6: Dependencias del plugin con otros proyectos.

proyectos de los que depende el plugin desarrollado. Los tres primeros _fwAndami, appgvSIG y libFmap son proyectos de los que dependen la mayoría deplugins. El resto son proyectos con dependencias particulares de este plugin. En concreto, extDielmoOpenLidar y libDielmoOpenLiDAR se tratan del driver de lectura y escritura de datos lidar desarrollado previamente a este proyecto por Dielmo 3D y una librería de apoyo a éste. En partes del código del plugin desarrollado se cargan capas LiDAR, y se hace uso de este driver. Por este motivo la dependencia con este proyecto. Por otra parte libDielmoLiDARAlgorithms, sextante y sextante_gui son proyectos que dan soporte a los algoritmos de análisis.

7.4.2 El fichero config.xml

Para comprender la estructura del pluguin es primordial la comprensión de su nexo de unión con gvSIG. Éste es el fichero de configuración config.xml situado en el subdirectorio config del plugin. Éste fichero plugin como se ha visto en el capítulo de análisis se trata de un fichero XML, es decir con estructura de arbol. En él se definen las extensiones que conforman el plugin, y para cada una de éstas una entrada o varias en la interfaz de usuario. En las siguientes secciones se detallan las distintas extensiones del plugin desarrollado en este proyecto donde se analizará, de forma particular, su porción de fichero config.xml y los elementos que lo componen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
    <libraries library-dir="."/>
    <depends plugin-name="com.iver.cit.gvsig"/>
```

```
<resourceBundle name="text"/>
<extensions>
    <extension class-name="com.dielmo.lidar.extension.LoadFromLiDARProjectExtension"
        description="Proyecto_Lidar"
        active="true">
        <menu text="LiDAR/load_LiDAR_blocks" tooltip="load_LiDAR_blocks"
            action-command="LOAD_LiDAR_BLOCKS"
            icon="images/lidarAddBlocks.png"
            position="1" />
        <tool-bar name="LiDAR_Tool">
            <action-tool icon="images/lidarAddBlocks.png"
                action-command="LOAD_LiDAR_BLOCKS" tooltip="load_LiDAR_blocks"
                position="1"/>
        </tool-bar>
    </extension>
    <extension class-name="com.dielmo.lidar.extension.ActiveProjectExtension"
        description="SeleccionProyectoActivo"
        active="true">
        <tool-bar name="LiDAR_Tool">
            <action-tool icon="images/lidarActive.png"
                action-command="LIDAR_DOC_ACTIVE" tooltip="lidar_document_active"
                position="3"/>
        </tool-bar>
        <menu text="LiDAR/lidar_document_active" tooltip="lidar_document_active"
            action-command="LIDAR_DOC_ACTIVE"
            icon="images/lidarActive.png"
            position="3" />
    </extension>
    <extension class-name="com.dielmo.lidar.extension.GraficaLongitudinalExtension"
        description="graf_long"
        active="true">
        <tool-bar name="LiDAR_Tool">
        <action-tool icon="images/longitudinal.png"
                action-command="grafica_long" tooltip="graf_long"
                position="4"/>
        </tool-bar>
        <menu text="LiDAR/graf_long" tooltip="graf_long"
            action-command="grafica_long"
            icon="images/longitudinal.png"
            position="4" />
    </extension>
    <extension class-name="com.dielmo.lidar.extension.InformePrecisionAlturaExtension"
        description="InformePrecision"
        active="true">
        <menu text="LiDAR/ControlCalidad/InformePrecision" tooltip="InformePrecision"
            action-command="InformePrecision"
            position="1"/>
    </extension>
    <extension class-name="com.dielmo.lidar.extension.AnalisisEjecucionVueloExtension"
        description="analisis_ejecucion"
        active="true">
        <menu text="LiDAR/ControlCalidad/analisis_ejecucion" tooltip="
            analisis_ejecucion"
            action-command="analisis_ejecucion"
            position="2"/>
    </extension>
    <extension class-name="com.dielmo.lidar.extension.AnalisisPrevioDatosExtension"
        description="AnalisisPrevioDatos"
        active="true">
        <menu text="LiDAR/ControlCalidad/AnalisisPrevioDatos" tooltip="
            AnalisisPrevioDatos"
```

```
            action-command="AnalisisPrevioDatos"
            position="0"/>
        </extension>
        <extension class-name="com.dielmo.lidar.extension.XYZtoLASExtension"
            description="Proyecto_Lidar"
            active="true">
            <menu text="Vista/Proyecto/XYZTOLAS_PAN" tooltip="XYZTOLAS"
                action-command="XYZTOLAS_PAN"
                icon="images/xyz_to_las.png" />
            <tool-bar name="LiDAR_Tool">
                <action-tool icon="images/xyz_to_las.png"
                    action-command="XYZTOLAS_PAN" tooltip="XYZTOLAS"
                    position="4"/>
            </tool-bar>
        </extension>
        <extension class-name="com.dielmo.lidar.extension.LAStoXYZWithFilterExtension"
            description="LasToXYZWithFilter"
            active="true">
            <tool-bar name="LiDAR_Tool">
                <action-tool icon="images/las_to_xyz.png"
                    action-command="XYZtoLAS" tooltip="las_to_xyz"
                    position="2"/>
            </tool-bar>
            <menu text="Vista/Proyecto/las_to_xyz" tooltip="las_to_xyz"
                action-command="las_to_xyz"
                icon="images/las_to_xyz.png" />
        </extension>
    </extensions>
</plugin-config>
```

7.4.3 La extensión de carga de capas

La extensión de carga de capas es la encargada de aportar la funcionalidad necesaria para la carga de capas cartgráficas en la vista definidas en el documento LiDAR activo. Para facilitar al técnico la carga de cartografía, éste únicamente debe realizar una selección de un rectángulo sobre una vista mediante una herramienta proporcionada por la extensión. A continuación aparecerá la cartografía definida en el documento LiDAR activo disponible en el área de selección, es decir, las capas cartográficas que intersectan con el rectángulo definido en la selección.

La extensión está definida en la siguiente porción del fichero config.xml:

```
<extension class-name="com.dielmo.lidar.extension.LoadFromLiDARProjectExtension"
          description="La extensión de carga de capas es la encargada de aportar la
              funcionalidad necesaria para la carga de capas cartgráficas en la vista
              definidas en el documento LiDAR activo"
          active="true">
          <menu text="LiDAR/load_LiDAR_blocks" tooltip="load_LiDAR_blocks"
              action-command="LOAD_LiDAR_BLOCKS"
              icon="images/lidarAddBlocks.png"
```

```
            position="1" />
        <tool-bar name="LiDAR_Tool">
            <action-tool icon="images/lidarAddBlocks.png"
                action-command="LOAD_LiDAR_BLOCKS" tooltip="load_LiDAR_blocks"
                position="1"/>
        </tool-bar>
    </extension>
```

La clase que extiende de la clase *Extension* es LoadFromLiDARProjectExtension. La etiqueta *description* sirve para una breve explicación de la utilidad de la extensión. Ésta aparecerá al seleccionar la extensión en el menú *preferencias/general/extensiones (ver imagen 7.7)*. La extensión declara un *menu text* y *un tool-bar* con un *action-tool icon*. El icono de ambos elementos de la interfaz de usuario es el mismo. Se trata del fichero lidarAddBlocks.png situado en la carpeta images del plugin. El *menu text* se corresponde con el menú LiDAR/cargador de capas LiDAR gracias a las siguientes entradas en el fichero de traducciones text.properties:

```
LiDAR=LiDAR
load_LiDAR_blocks=Cargador de capas LiDAR
```

La traducción del valor de la etiqueta *tooltip* aparecerá al situar el cursor sobre el elemento de la interfaz de usuario (ver imagen 7.8).

La extensión está inicialmente activa, aunque no aparece si la ventana activa no se trata de una vista, esto se debe al metodo *isVisible()* de la clase *LoadFromLiDARProjectExtension*, y además no estará habilitada a menos que exista algún documento LiDAR creado, esto se debe al método isEnabled(). Este comportamiento se debe al código siguiente código:

```
public boolean isEnabled() {
        ProjectExtension projectextension = (ProjectExtension) PluginServices.getExtension(
            com.iver.cit.gvsig.ProjectExtension.class);
        ArrayList lidarProjects = projectextension.getProject().getDocumentsByType(
            LiDARDocumentFactory.registerName);
        if(lidarProjects.size()!=0) return true;
        else return false;
    }

public boolean isVisible() {
        IWindow window = PluginServices.getMDIManager().getActiveWindow();
        if (window instanceof View) return true;
        return false;
    }
```

El action-command no es un parámetro importante en este caso ya que la extensión sólo ejecuta una funcionalidad y éste sirve precisa-

Figura 7.7: Descripción de la extensión en el menú preferencias.

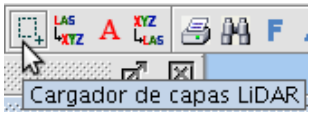


Figura 7.8: Descripción de la extensión en el menú preferencias.

mente para discriminar distintas acciones dentro de una misma extensión. Se trata de un parámetro que se pasa al invocar al metodo *execute()* al pulsar en un elemento visible y activo de la interfaz de usuario declarado en la extensión, en este caso al pulsar sobre el *menu text* o el *un tool-bar*. Por tanto el punto de entrada desde gvSIG al código desarrollado en este proyecto es a través de este método de cada una de las extensiones implementadas.

En el caso de esta extensión, el método *execute* se trata del siguiente código:

```
public void execute(String actionCommand) {
       if(actionCommand.equals("LOAD_LiDAR_BLOCKS")){

           View view = (View)PluginServices.getMDIManager().getActiveWindow();
           // escuchamos los rectangulos.
           if(view == null)
               return;
           MapControl mc = view.getMapControl();
           //Listener de eventos de movimiento que pone las coordenadas del ratón en la
               barra de estado
           StatusBarListener sbl = new StatusBarListener(mc);
           //Seleccionar capas
           listener = new com.dielmo.lidar.document.SelectLayersListener(mc);
           mc.addMapTool("selectLayers", new Behavior[]{
                  new RectangleBehavior(listener), new MouseMovementBehavior(sbl)});

           mc.setTool("selectLayers");
       }

   }
```

Se obtiene el objeto *MapControl* de la vista, que se corresponde con

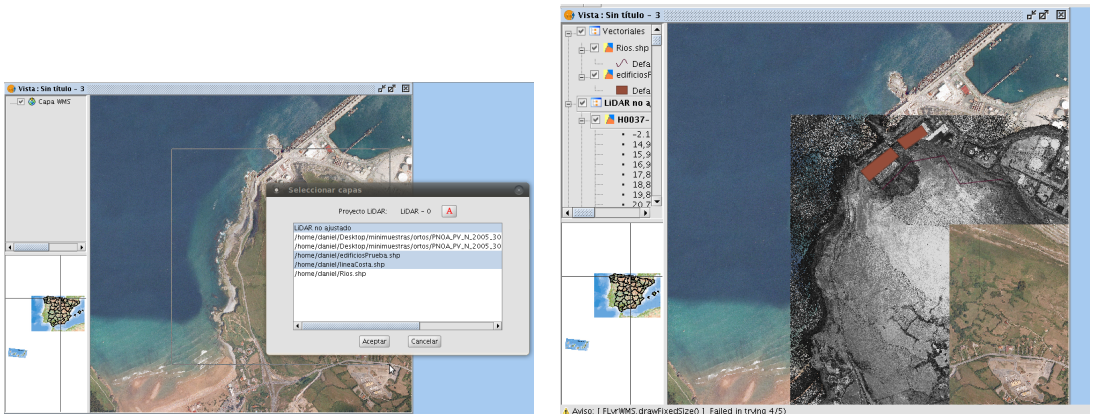


Figura 7.9: Herramienta de cargado automático de capas.

la ventana activa (era una restricción impuesta en el método *isVisible*()) y se le añade una herramienta que permite realizar rectángulos sobre la vista disparando el evento SelectLayersListener tras realizarlo. En éste método se realizan las acciones necesarias para obtener el rectángulo dibujado en el mapa. Tras ello traducir la selección a coordenadas del mundo real y comparar con toda la cartografía que usuario haya definido en el documento LiDAR activo. Así, finalmente, poder ofrecer la posibilidad de cargar las capas cartográficas definidas que intersecten con esa zona seleccionada. Este proceso se muestra en la secuencia de imágenes 7.9.

El método Java que calcula la extensión de una capa está ubicado en la clase Producto.java. Esta clase ha sido creada para definir todas las capas cartográficas en un proyecto LiDAR como un objeto de tipo producto. Los atributos que tiene un producto son 5: name, leyenda, driverName, extension, path. name es el nombre que identifica el tipo de producto, es decir, si se trata de un fichero lidar ajustado, una ortofoto, un vectorial, etc., *leyenda* es un String que contiene la ruta absoluta al fichero de leyenda definida. Ésta se utiliza en el PanelVectorialesOption.java para asignar leyendas .gvl a las capas vectoriales. *driverName* es un String con el mbre del driver que usara el producto al cargarse, *path* es la ruta absoluta al fichero en cuestión y finalmente *extension* es un objeto de tipo Rectangle2D que representa la extensión de la capa al ser cargada en una vista. Este último atributo permite la implementación del cargador automático de capas. Con este atributo, rectángulo, se compara el rectángulo definido por el usuario,

tras seleccionar la herramienta de la barra de herramientas para carga automática de capas y pintarlo sobre la vista. Por lo tanto la extensión de carga de capas busca, en los productos definidos en el documento LiDAR activo cuales están disponibles para la selección realizada.

A continuación se muestra ,en primer lugar, el código Java que calcula el Rectangle2D asociado a un Producto, y en segundo lugar, el código que compara la extensión de una colección de capas LiDAR ajustadas, con el rectángulo de selección de la herramienta, para verificar si son de la misma zona:

```
/**
 * Metodo que calcula el fullextent del pruducto y lo setea
 * Unicamente ha de tener seteado el path como requisito
 * Aqui NO setearemos el campo Name ya que es generico y no se si estoy tratando lidar
     ajustados,
 * sin ajustar, ortos, vectoriales, AOI o Bloques
 */
public void calcularExtension(){

    String path =this.path;
    File f = new File(path);
    FiltroTipoArchivos filtro = new FiltroTipoArchivos();
    if(path.equals("")){
        //si no tiene una ruta la extension la seteamos a 0
        this.setExtension(new Rectangle2D.Double());
    }
    else if(filtro.esVectorial(f)){
        this.setDriverName(filtro.getVectorialDriverName(f));
        Rectangle2D rect=null;

        try {
            Driver d = LayerFactory.getDM().getDriver(this.getDriverName());
            ((VectorialFileDriver)d).open(f);
            ((VectorialFileDriver)d).initialize();

            // comprueba que contenga datos el fichero.
            if(((VectorialFileDriver)d).getShapeCount()>0){

                // obtenemos el rectagle2D que forman los vectoriales y lo asignamos al
                    producto
                rect = ((VectorialFileDriver)d).getFullExtent();
                this.setExtension(rect);
            }
            else
                this.setExtension(new Rectangle2D.Double());
        }catch (IOException exc) {
                // TODO Auto-generated catch block
                JFrame j= new JFrame();
                JOptionPane.showMessageDialog(j,PluginServices.getText(null,"Error al
                    cargar el fichero"+this.getPath()));
                this.setExtension(new Rectangle2D.Double());
                exc.printStackTrace();
        }
```

```
        }
        else if (filtro.esImagen(f)){
            this.setDriverName("gvSIG Image Driver");
            Rectangle2D rect=null;
            try {
                FLayer layer;
                layer = LayerFactory.createLayer("auxiliar", this.driverName,
                        f, Project.getDefaultProjection());

                if(layer!=null && layer.isOk()){

                    // obtenemos el rectagle2D que forman los shp
                    rect = layer.getFullExtent();
                    this.setExtension(rect);
                }
            }catch(Exception ex){
                JFrame j= new JFrame();
                JOptionPane.showMessageDialog(j,PluginServices.getText(null,"Error al
                    cargar el fichero"+this.getPath()));
                this.setExtension(new Rectangle2D.Double());
                ex.printStackTrace();
            }
        }
    }
```

Código que compara la extensión de una colección de capas LiDAR ajustadas, con el rectángulo de selección de la herramienta:

```
//la variable rect contiene el rectángulo de selección del usuario sobre la vista
...
for(int i=0;i<productosLiDARAdjusted.size();i++){
                if(productosLiDARAdjusted.get(i).getExtension().intersects(rect)){
                    if(numProductosLiDARAdjustedIntersect==0)
                        model.addElement(productosLiDARAdjusted.get(0).getName()); //todos
                            se llaman igual, cojo cualquiera
                    numProductosLiDARAdjustedIntersect++;
                    //corto el bucle cuando encuentro 2 que lo intersectan ya que si solo
                        hay uno que intersecta
                    //no crearé una agrupacion de capas y si hay mínimo 2 sí la crearé.
                    if(numProductosLiDARAdjustedIntersect==2) break;
                }
            }
```

7.4.4 La extensión de selección del proyecto activo

Esta extensión permite la selección del documento LiDAR activo. És necesario definir uno como activo para ciertas operaciones como el caso de la extensión de carga automática de capas definidas en un documento LiDAR. Puesto que es posible tener definidos varios documentos LiDAR al mismo tiempo se necesita saber de cual de ellos obtener la información.

```
<extension class-name="com.dielmo.lidar.extension.ActiveProjectExtension"
           description="SeleccionProyectoActivo"
           active="true">
```

Figura 7.10: Botón de la extensión de selección del documento LiDAR activo.

```
            <tool-bar name="LiDAR_Tool">
                <action-tool icon="images/lidarActive.png"
                    action-command="LIDAR_DOC_ACTIVE" tooltip="lidar_document_active"
                    position="3"/>
            </tool-bar>
            <menu text="LiDAR/lidar_document_active" tooltip="lidar_document_active"
                action-command="LIDAR_DOC_ACTIVE"
                icon="images/lidarActive.png"
                position="3" />
        </extension>
```

En el caso de otros documentos, como por ejemplo las vistas, se considera activa aquella que tiene el foco. Sin embargo en el caso de un documento LiDAR no se interactúa con su interfaz, sino que se definen los datos y se cierra el panel del documento, y estos datos serán leídos en interacciones con vistas, en algoritmos de cálculo, etc, de ahí la importancia de esta extensión por definir el documento LiDAR activo.

De la misma forma que en la extensión anterior, se han declarado un *action-tool* y un *menu text*, es decir, una entrada en el mismo menú que la extensión anterior y un botón en la barra de herramientas de la ventana principal de gvSIG. En la figura 7.10 se muestra la apariencia del botón de la barra de herramientas y el tooltip resultado de traducir el valor la etiqueta *tooltip* del xml de la extensión.

A continuación se muestra el trozo de código Java que implementa el botón de aceptar tras seleccionar el documento activo que dispone de la funcionalidad necesaria para setearlo como tal:

```
private JButton getJButtonAccept(){

        JButton jButtonAcetpar=new JButton(PluginServices.getText(null,"Aceptar"));
        jButtonAcetpar.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                if(jComboLidarProjects.getSelectedItem()!= null){

                    lidarDocumentSelected = (LiDARDocument)jComboLidarProjects.
                        getSelectedItem();
                    LiDARProject.setInstance(lidarDocumentSelected.getLidarProject());
                    lidarDocumentSelected.getLidarProject().setSoyProyectoActivo(true);
```

```
                //recorro los demas y los seteo a false
                ProjectExtension projectextension = (ProjectExtension) PluginServices.
                    getExtension(com.iver.cit.gvsig.ProjectExtension.class);
                lidarProjects = projectextension.getProject().getDocumentsByType(
                    LiDARDocumentFactory.registerName);
                 for(int i = 0;i<lidarProjects.size();i++){
                    LiDARDocument ld=lidarProjects.get(i);
                    if(!ld.getLidarProject().equals(lidarDocumentSelected.
                        getLidarProject()))
                       lidarProjects.get(i).getLidarProject().setSoyProyectoActivo(
                           false);
                 }

            }
            PluginServices.getMDIManager().closeWindow(LidarDocumentSelectionPanel.this
                );

        }
    });

    return jButtonAcetpar;
}
```

7.4.5 La extensión de visualización de gráficas longitudinales

Esta extensión permite la visualización de perfiles longitudinales de los datos LiDAR. Esta funcionalidad, como se ha mostrado durante el análisis de otras aplicaciones que utilizan datos LiDAR, es muy útil para un análisis visual de los datos.

```
<extension class-name="com.dielmo.lidar.extension.GraficaLongitudinalExtension"
          description="Creación de gráficas longitudinales a partir de la selección de
              una franja sobre datos LiDAR. "
          active="true">
          <tool-bar name="LiDAR_Tool">
          <action-tool icon="images/longitudinal.png"
                 action-command="grafica_long" tooltip="graf_long"
                 position="4"/>
          </tool-bar>
          <menu text="LiDAR/graf_long" tooltip="graf_long"
              action-command="grafica_long"
              icon="images/longitudinal.png"
              position="4" />
      </extension>
```

Para la implementación de la herramienta se ha declarado un *action-tool* y un *menu text*, una entrada en el mismo menú que la extensión anterior y un botón en la barra de herramientas de la ventana principal de gvSIG.

```
public class GraficaLongitudinalExtension extends Extension {

    private com.dielmo.lidar.utiles.chart.GraficaLongitudinalLineListener listener = null;
```

```
    public void execute(String actionCommand) {
        if(actionCommand.equals("grafica_long")){
            View view = (View)PluginServices.getMDIManager().getActiveWindow();
            if(view == null) return;
            MapControl mc = view.getMapControl();
            StatusBarListener sbl = new StatusBarListener(mc);
            listener = new com.dielmo.lidar.utiles.chart.GraficaLongitudinalLineListener(mc
                );
            mc.addMapTool("grafica_long", new Behavior[]{
                new ThreePointsBehavior(listener), new MouseMovementBehavior(sbl)});
            mc.setTool("grafica_long");
        }
    }

    public boolean isEnabled() {
        com.iver.andami.ui.mdiManager.IWindow f = PluginServices.getMDIManager().
            getActiveWindow();
        if (f == null) return false;
        if (f instanceof View) {
            View vista = (View) f;
            IProjectView model = vista.getModel();
            FLayers layers = model.getMapContext().getLayers();
            FLayer lyr;
            SingleLayerIterator it = new SingleLayerIterator(layers);
            while (it.hasNext()) {
                lyr = it.next();
                if(LiDARDriver.isLidarLayer(lyr) && lyr.isAvailable()) return true;
            }
        }
        return false;
    }

    public boolean isVisible() {

        com.iver.andami.ui.mdiManager.IWindow f = PluginServices.getMDIManager().
            getActiveWindow();
        if (f == null) return false;
        if (f instanceof View) {
            View vista = (View) f;
            IProjectView model = vista.getModel();
            FLayers layers = model.getMapContext().getLayers();
            FLayer lyr;
            SingleLayerIterator it = new SingleLayerIterator(layers);
            while (it.hasNext()) {
                lyr = it.next();
                if(LiDARDriver.isLidarLayer(lyr) && lyr.isAvailable()) return true;
            }
        }
        return false;
    }

    public void initialize() {}

    %%%%%%%%%%
}
```

En el código de la extensión se observa que los métodos *isVisible*() y *isEnabled*(), que determinan la visibilidad y disponibilidad de las her-

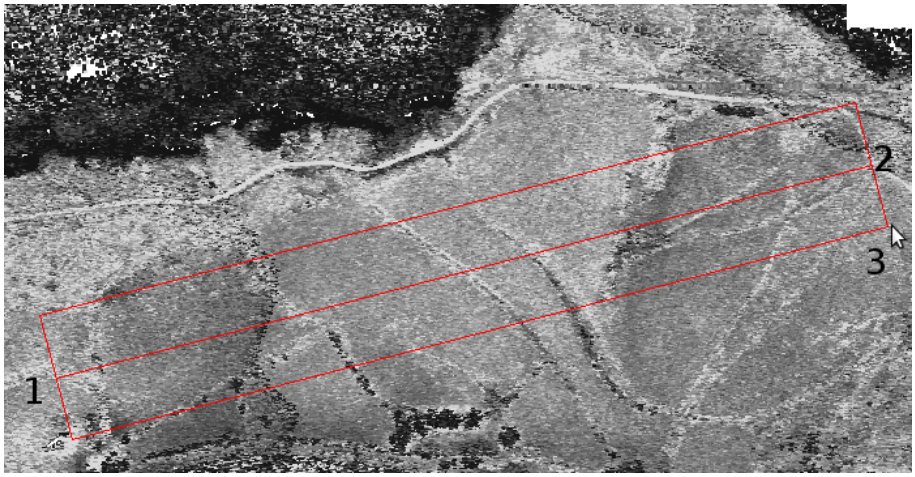


Figura 7.11: Herramienta de selección de franjas.

ramientas del menú y de la barra, permitirán el uso de la herramienta de perfiles longitudinales siempre que la ventana activa sea una vista que contenga alguna capa LiDAR disponible. De la misma forma que en la extensión de carga de capas automática, el método *execute* activa una herramienta que permite interactuar con la vista. Esta herramienta permite localizar 3 puntos que definen la franja a mostrar el perfil. Los dos primeros puntos forman una línea y el tercero define el grosor de la franja (ver figura 7.11).

A continuación se muestra la parte de código java que calcula la franja a partir de los tres puntos definidos:

```
/**
     * Obtiene la IGeometry que forman los 3 puntos con los que formamos nuestra franja
         rectangular de seleccion de puntos LiDAR
     * en la vista.
     *
     * @param franja Franja de la cual extraemos la geometria.
     * @return geometria que forma el rectangulo de nuestra franja de entrada.
     */
    private IGeometry obtenerIGeometryFranja(Franja franja){

        Point2D p1=franja.getPunto1();
        Point2D p2=franja.getPunto2();
        Point2D p3=franja.getPunto3();

        org.jfree.data.xy.Vector vec2 = new org.jfree.data.xy.Vector(p2.getY() - p1.getY()
            ,-(p2.getX() - p1.getX()) );
        double moduleV2=vec2.getLength();
        org.jfree.data.xy.Vector vec2Unit = new org.jfree.data.xy.Vector(vec2.getX()/
            moduleV2 , vec2.getY()/moduleV2 );
        Line2D l=new Line2D.Double(p1,p2);
        grosor=2*l.ptLineDist(p3);
```

```
        //busco punto pvert1 en recta y=m2x + n2 q ademas diste GROSOR/2 de p1
        //la distancia es el modulo del vectorr entre ambos puntos
        Point2D pvert1 = new Point2D.Double(p1.getX()+vec2Unit.getX()*(grosor/2) , p1.getY
            ()+vec2Unit.getY()*(grosor/2));
        Point2D pvert2 = new Point2D.Double(p1.getX()-vec2Unit.getX()*(grosor/2) , p1.getY
            ()-vec2Unit.getY()*(grosor/2));
        Point2D pvert3 = new Point2D.Double(p2.getX()+vec2Unit.getX()*(grosor/2) , p2.getY
            ()+vec2Unit.getY()*(grosor/2));
        Point2D pvert4 = new Point2D.Double(p2.getX()-vec2Unit.getX()*(grosor/2) , p2.getY
            ()-vec2Unit.getY()*(grosor/2));

        // obtenemos la geometria con los puntos que formaran el rectangulo de nuestra
            Franja de seleccion sobre la capa LiDAR.
        GeneralPathX iteratorPath = new GeneralPathX();
        iteratorPath.moveTo(pvert1.getX(),pvert1.getY());
        iteratorPath.lineTo(pvert2.getX(),pvert2.getY());
        iteratorPath.lineTo(pvert4.getX(),pvert4.getY());
        iteratorPath.lineTo(pvert3.getX(),pvert3.getY());
        iteratorPath.lineTo(pvert1.getX(),pvert1.getY());

        IGeometry geom = ShapeFactory.createPolygon2D(iteratorPath);

        return geom;
    }
```

Una vez obtenida la franja del terreno que se desea representar mediante un perfil, en primer lugar hay que obtener qué puntos lidar, están dentro de esta franja (si desatendemos la altura z del punto LiDAR el punto está situado dentro del rectámgulo definido por la franja), en segundo lugar los puntos hay que trasladarlos al origen de coordenadas y finalmente rotarlos para poder visualizarlos de frente mediante sus alturas. A continuación se adjunta la parte de código que se encarga de esta tarea:

```
//obtengo la IGeometry de la franja de seleccion y creo el objeto TransformaPuntos3D para
    trasladar y rotar puntos
        IGeometry geometry= obtenerIGeometryFranja(franja);

        //para trasladar y rotar los puntos inicializo trans3d
        trans3d = new TransformaPuntos3D(franja.getPunto1().getX(),franja.getPunto1().getY
            (),franja.getPunto2().getX(),franja.getPunto2().getY(),null);

        //arraylist de todas las capas lidar visibles en la vista
        ArrayList<FLyrVect> capasLidarVisibles = obtenerCapasLidarVisibles();

        ArrayList<double[]> todosLosRows=new ArrayList<double[]>();

        for(int i=0;i<capasLidarVisibles.size();i++){

            FLyrVect lyrVect= capasLidarVisibles.get(i);
            if(LiDARDriver.isLidarLayer(lyrVect)){
                ArrayList<double[]> dat = getArrayPuntosFromGeom(geometry,lyrVect, i);
                todosLosRows.addAll(dat);
            }
        }

        // seteamos los row de puntos
```

```
        this.trans3d.setPuntos(todosLosRows);
        float[][] data=transformaPuntos(todosLosRows);

        capaSeleccionXZ = crearCapaAuxiliar(todosLosRows);

        //  la capa auxiliar contiene los puntos dentro de la franja ya trasladados y
            rotados.

        // Pintamos la gráfica en un diagrama basado en un panel de jfreechart adaptado
            para la funcionalidad erquerida.
       ScatterPlotDialog panel2= new ScatterPlotDialog(565,400,data, trans3d, grosor,
           capaSeleccionXZ);
```

En el código anterior hay que aclarar un par de métodos. *getArrayPuntosFromGeom*() y *transformaPuntos*(). El primero de ellos *getArrayPuntosFromGeom* realiza una petición queryByShape o lo que es lo mismo obtiene los puntos situados dentro de la geometría pasada como parámetro.

```
    /**
     * Obtiene los rows de la capa flyer que intersectan con la geometria geom y fueron
         pintados en la vista.
     *
     * @param geom geometria de interes para la seleccion de puntos.
     * @param flyer capa sobre la que extraer los puntos.
     * @param iDLayer id del orden de las capas visibles en el toc
     * @return array con los row de los puntos que buscamos dentro de la capa y que
         intersectan con la geometria.
     */
    private ArrayList<double[]> getArrayPuntosFromGeom(IGeometry geom, FLyrVect flyer, int
         iDLayer){

        ArrayList<double[]> arrayRetornar = null;
        if(flyer.getStrategy() instanceof LiDARStrategy){
            LiDARStrategy lidarStrategy = (LiDARStrategy) flyer.getStrategy();

            ArrayList<Integer> indicesVista = (ArrayList<Integer>) lidarStrategy.getIndice
                ().clone();
            if(indicesVista!=null) arrayRetornar = queryByShape(geom, flyer, indicesVista,
                iDLayer);
        }
        return arrayRetornar;
    }
```

El segundo de los métodos transformaPuntos traslada y rota punto por punto usando la clase TransformaPuntos3D como se muestra a continuación:

```
        /**
     * Transforma los puntos LiDAR que caen dentro de nuestra franja.
     *
     * @param arrayDatos vector de puntos
     *
     * @return matriz con las xz transformadas de todos los puntos.
     */
    private float[][] transformaPuntos(ArrayList<double[]> arrayDatos){
```

```
        float[][] data = new float[3][arrayDatos.size()];

        for(int i=0;i< arrayDatos.size();i++){
            double[] punto =arrayDatos.get(i);
            this.trans3d.trasladaPunto(punto);
            this.trans3d.rotaPunto(punto);

            // nos quedamos con la x y la z para hacer la grafica respecto a esos ejes.
            data[0][i]= (float)punto[0];
            data[1][i]= (float)punto[2];
            data[2][i]= (float)punto[5];
        }

        return data;
    }
```

```
    /**
     * Traslada el row pasado, el cual contiene en la posicion 0 la x y en la posicion 1
         la y.
     * La traslacion consta a restarle la x e y del primero punto con el cual formabamos
         nuestra franja de seleccion.
     *
     * @param punto punto a trasladar.
     */
    public void trasladaPunto(double[] punto){

        punto[0]=punto[0]-this.x1;
        punto[1]=punto[1]-this.y1;

    }

    /**
     * Rota el row pasado, el cual contiene en la posicion 0 la x y en la posicion 1 la y.
     * La rotacion consta en la formula x*cos(angulo)-y*sin(angulo), siendo angulo el
         angulo que forma la recta de la franja
     * con respecto al eje x.
     *
     * @param punto punto a rotar.
     */
    public void rotaPunto(double[] punto){

        double x=punto[0];
        double y=punto[1];

        punto[0] = x*Math.cos(this.angulo) - y*Math.sin(this.angulo);
        punto[1]= x*Math.sin(this.angulo) + y*Math.cos(this.angulo);
    }
```

El resultado ver figura 7.12.

7.4.6 La extensión de conversión de .LAS a .XYZ

```
<extension class-name="com.dielmo.lidar.extension.LAStoXYZWithFilterExtension"
           description="LasToXYZWithFilter"
           active="true">
           <tool-bar name="LiDAR_Tool">
               <action-tool icon="images/las_to_xyz.png"
                   action-command="XYZtoLAS" tooltip="las_to_xyz"
                   position="2"/>
           </tool-bar>
```

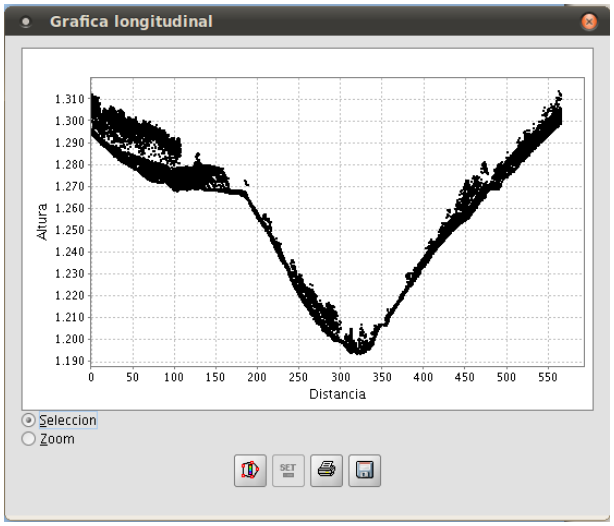


Figura 7.12: Perfil de un valle.

```
        <menu text="Vista/Proyecto/las_to_xyz" tooltip="las_to_xyz"
            action-command="las_to_xyz"
            icon="images/las_to_xyz.png" />
    </extension>
```

La extensión de conversión de formatos de .LAS a XYZ dota a gvSIG de la capacidad de pasar a distintos formatos los datos para asegurar su compatibilidad con otros softwares. Permite en primer lugar agregar una cantidad de ficheros a la herramienta de conversión, lo que permitirá lanzar un único proceso que realice todas las conversiones. En segundo lugar dispone de un filtro que permite realizar expresiones a ser evaluadas por cada uno de los puntos LiDAR, pudiendo así únicamente pasar al formato destino, por ejemplo, los puntos en un rango de alturas determinado, de una intensidad determinada, o los obtenidos entre un determinado ángulo de apertura del láser. En tercer lugar permite definir los ficheros de salida, pudiendo decidir qué campos de entrada serán campos de salida, en qué orden situar estos campos, qué precisión en decimales tendrán, y si se quiere obtener un único fichero de salida o varios de un tamaño máximo definido. Se trata por tanto de una herramienta avanzada de conversión. Una vez definidos todos los parámetros se lanza la tarea en segundo plano haciendo uso del servicio que proporciona andami a los plugins para este fin.

Por código se abren las capas LiDAR para ver los campos disponibles,y así mostrarlos para permitir realizar una expresión de filtro utilizando

los campos en ella. A continuación se muestra el trozo de código que abre una capa LiDAR y verifica los campos que tiene:

```
...
Driver d = LayerFactory.getDM().getDriver("gvSIG LIDAR driver");
if (result == JFileChooser.APPROVE_OPTION && (files = fileChooser.getSelectedFiles()) !=
    null) {
   for(int i=0; i< files.length; i++){
       String s =files[i].getAbsolutePath();

           try {
               ((VectorialFileDriver)d).open(new File(s));
               ((VectorialFileDriver)d).initialize();
               FLyrVect lyrFirst = (FLyrVect) LayerFactory.createLayer("capa"+ncapa,"gvSIG
                    LIDAR driver", new File(s),Project.getDefaultProjection());
               ncapa++;
               lyrFirst.getRecordset().getFieldNames();
               numLidarFormat=InicializeLidar.testFormat(new File(s));
               lidarFilesModel.addElement(s);
               arrayFlyerVectSeleccion.add(lyrFirst);

           catch (Exception e1) {e1.printStackTrace();}
...
```

Para filtrar los datos según la expresión de filtrado se utiliza una sentencia sql que se ha confeccionado concatenando condiciones según introduzca el usuario. Una vez evaluada la expresión, los puntos que cumplan las condiciones solicitadas serán los que se convertirán de formato. El código que realiza la selección de los puntos:

```
//en expr2 tenemos las condiciones que se han indicado en el filtro del panel que deben
    cumplir los puntos a convertir
final String expr = "select * from '" +
   model.getDataSourceName() + "' where " + expr2 + ";";
   logger.debug(expr);

   for (int i = 0; i < expressionListeners.size(); i++) {

       ExpressionListener expresion = (ExpressionListener) expressionListeners.get(i);
       //realiza la consulta, si no hay consulta selecciona todos los puntos
       expresion.newSet(expr);

   }
```

Parte del código que lanza la tarea de conversión en segundo plano:

```
...
LiDARToXYZMonitorableTask lmt = new LiDARToXYZMonitorableTask(LAStoXYZFilterPanel.this);
PluginServices.cancelableBackgroundExecution(lmt);
PluginServices.getMDIManager().closeWindow(LAStoXYZFilterPanel.this);


...
```

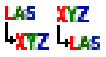


Figura 7.13: Iconos .LAS a .XYZ. y .XYZ a .LAS

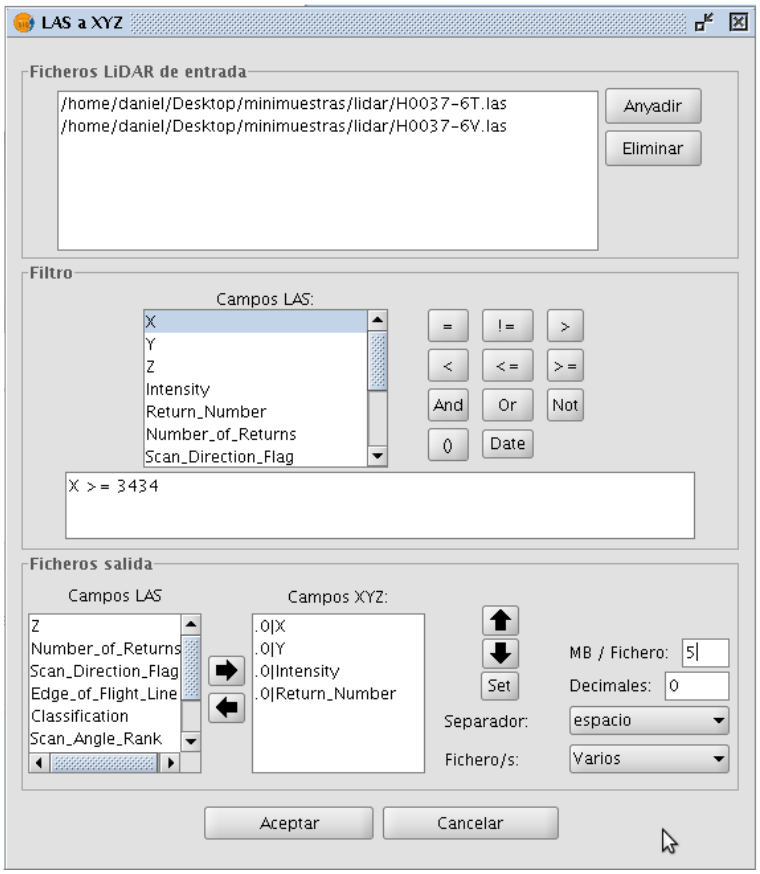


Figura 7.14: Herramienta conversión de .LAS a .XYZ.

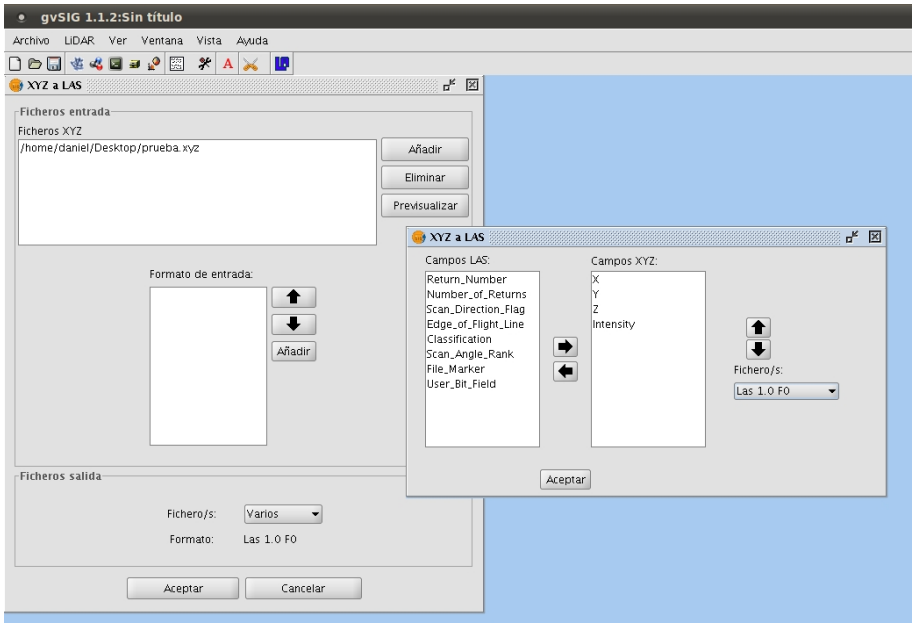


Figura 7.15: Herramienta conversión de .XYZ a .LAS

7.4.7 La extensión de conversión de .XYZ a .LAS

Esta extensión es idéntica a la anterior en cuanto a la configuración del config.xml y en cuanto a implementación de los métodos *isVisible*(), *isEnabled*(). La funcionalidadn es la contraria a la extensión anterior. Permite seleccionar el formato .LAS de salida.

```
<extension class-name="com.dielmo.lidar.extension.XYZtoLASExtension"
    description="Proyecto_Lidar"
    active="true">
    <menu text="Vista/Proyecto/XYZTOLAS_PAN" tooltip="XYZTOLAS"
        action-command="XYZTOLAS_PAN"
        icon="images/xyz_to_las.png" />
    <tool-bar name="LiDAR_Tool">
        <action-tool icon="images/xyz_to_las.png"
            action-command="XYZTOLAS_PAN" tooltip="XYZTOLAS"
            position="4"/>
    </tool-bar>
</extension>
```

En cuanto a implementación se refiere, se trata de una extensión más sencilla que la anterior. Esto se debe a que el formato de entrada es un fichero de texto .XYZ con un punto en cada línea cuyos campos aparecen separados por un separador que puede ser diferente en distintos ficheros (coma, punto tabulador) y disponer de campos distintos en cada ocasión. La escritura en formato .LAS, sin embargo, se realiza utilizando el driver de escritura para LiDAR que Dielmo había desarrollado anterior a este proyecto. A continuación se muestra una

parte de código que realiza esta función:

```
...
    IWriter m_Writer = new LiDARWriter();
    ...
    File fileLAS = new File(s);
    ((LiDARWriter)m_Writer).setFile(fileLAS);

    //Para definir el tipo de punto en función del formato
...
    if(jlabelFormatoLas.getText().equals("Las 1.0 F0")){
        header = new LASHeader_V10(null);
        ((LASHeader_V10)(header)).SetPointDataFormatID((byte)0);
        point = new LASPoint10F0();
        }
        else if(jlabelFormatoLas.getText().equals("Las 1.0 F1")){
            header = new LASHeader_V10(null);
            ((LASHeader_V10)(header)).SetPointDataFormatID((byte)1);
            point = new LASPoint10F1();
            }
.....

        header.setXScale(0.01);
        header.setYScale(0.01);
        header.setZScale(0.01);
        header.setXOffset(0);
        header.setYOffset(0);
        header.setZOffset(0);

        ((LiDARWriter)m_Writer).setHeaderAndPoint(header, point);
        ContainerColumnDescription cols = new ContainerColumnDescription();
        point.getColumnsDescription(cols);
        FieldDescription[] fields =getFields(point);
        tableDef.setFieldsDesc(fields);
        tableDef.setName(s);

        try {
        //para iniciar el writer
            m_Writer.initialize(tableDef);
            m_Writer.preProcess();
            ....

            Value[] v = new Value[fields.length];
                        for(int t=0;t<v.length;t++){

                            if(point.getColumnsDescription(cols).get(t).getFieldType()==
                                ColumnDescription.DOUBLE)
                               v[t]=ValueFactory.createValue(0.0);
                            else if(point.getColumnsDescription(cols).get(t).getFieldType
                                ()==ColumnDescription.INT)
                               v[t]=ValueFactory.createValue(0);

                            else if(point.getColumnsDescription(cols).get(t).getFieldType
                                ()==ColumnDescription.BYTE){
                               byte a=0;
                               v[t]=ValueFactory.createValue(a);

                            }
                            else
                               v[t]=ValueFactory.createValue(0);
                        }
```

```
String sFile=lidarFilesModel.get(i).toString();
FileReader fr = new FileReader (sFile);
BufferedReader br = new BufferedReader(fr);

StringTokenizer t1 = new StringTokenizer(linea,separador);
int totalTokens=t1.countTokens();

    //iterar lineas y setear los valores del punto

    //leer cada linea
    for(int t=0;t<totalTokens;t++){
        String tok =t1.nextToken();
        if(point.getColumnsDescription(cols).get(indices[t]).
            getFieldType()==ColumnDescription.DOUBLE)
            v[indices[t]]=ValueFactory.createValue(Double.
                parseDouble(tok));
        else if(point.getColumnsDescription(cols).get(indices[t])
            .getFieldType()==ColumnDescription.INT){
            double d = Double.parseDouble(tok);
            int in= (int)d;
            v[indices[t]]=ValueFactory.createValue(in);
        }
        else if(point.getColumnsDescription(cols).get(indices[t])
            .getFieldType()==ColumnDescription.BYTE){
            Byte b = new Byte(tok);
            v[indices[t]]=ValueFactory.createValue(b);
        }
        else{
            int y=0;
            y=y+1;
        }

        //la geometria, es decir, el punto
    IGeometry geom=ShapeFactory.createGeometry(new FPoint2D(new
        Double(v[0].toString()).doubleValue(), new Double(v[1].
        toString()).doubleValue()));
    IRow feat=new DefaultFeature(geom,v,new String(""+numero) );
    IRowEdited rowEdited=new DefaultRowEdited(feat, IRowEdited.
        STATUS_ADDED, numero);
    m_Writer.process(rowEdited);

    }

    ....
    //al finalizar
    m_Writer.postProcess();
```

7.5 Implementación de algoritmos

- **previewxyzdielmoopenlidar**. Se trata de un algoritmo que previsualiza los datos de un fichero xyz.

- **readbinheaderdielmoopenlidar** Previsualiza la cabecera de un fichero BIN.

- **agrupapuntosdielmoopenlidar** Este algoritmo coge como entrada una capa vectorial en formato shp de puntos y una distancia en metros (por defecto 50m). El resultado es una nueva capa vectorial de puntos en formato shp que contiene la misma información que la capa de entrada, al que se le ha añadido un campo nuevo con un identificador de grupo. Todos aquellos puntos cuya distancia entre alguno de ellos sea inferior a la distancia indicada en el campo de entrada se clasificarán con el mismo número de grupo. Los grupos se numeran por orden correlativo desde el nº 1 hasta el número de grupos encontrado en el fichero de entrada.
- **analisisejecucionvuelodielmoopenlidar** Este algoritmo recorre el 100 % de los puntos LiDAR apoyándose en imágenes de la resolución de trabajo indicada. Cuanto menor sea la resolución de trabajo, se obtendrá una mayor definición en la forma de los polígonos obtenidos como ficheros de salida, pero por el contrario llevará más tiempo de cálculo y si la extensión de las capas LiDAR a la resolución de trabajo no cabe en memoria se producirá un error. Lo ideal para el formato que tienen los datos LiDAR del PNOA es trabajar a una resolución de 20x20m. Como resultados se obtienen 4 ficheros vectoriales:
  - *Zona volada*: Mapa vectorial de polígonos con la zona volada, almacenando un polígono por cada línea de vuelo presente en cada uno de los ficheros LAS que componen el vuelo LiDAR. Además para cada uno de los polígonos se almacena el código de línea de vuelo y el número de puntos LiDAR que contiene.
  - *Agujeros*: Mapa vectorial de polígonos con las zonas que han quedado sin volar. Estas zonas sin dato suelen ser zonas de agua, pero este análisis nos puede ayudar a detectar si hay alguna zona que se ha quedado sin volar debido a un error del sensor. En este algoritmo se obtiene una capa provisional que sirve como entrada al algoritmo siguiente para obtener el fichero de agujeros definitivo.
  - Densidades: Mapa vectorial de polígonos con la densidad del número de puntos por metro cuadrado con un paso de malla de 100m en formato vectorial. Este mapa nos permitirá analizar si se han cumplido los requerimientos de densidad de puntos por metro cuadrado.

- Puntos planos: A partir de los datos LiDAR originales, buscamos zonas muy planas, donde la variación máxima en altura (diferencia entre la altura máxima y mínima) sea inferior a 30cm dentro de la resolución de trabajo. A partir de esta selección de puntos, nos quedaremos con aquellos que caen en zonas de solape y almacenamos un fichero vectorial de puntos con las diferencias de alturas entre pasadas que nos servirá para realizar un análisis del ajuste en altura entre pasadas.

- **analisisejecucionvuelo2dielmoopenlidar** Este algoritmo es la continuación del algoritmo analisis de ejecución del vuelo, y coge como entrada las salidas descritas en dicho algoritmo. Se ha decidido separar este proceso en dos fases debido a que la primera fase tiene un elevado coste computacional a tener que recorrer el 100 % de los datos, y si se produce un error en esta segunda fase no es necesario tener que repetir todo el proceso anterior. Como resultado se obtienen otros 4 ficheros vectoriales que describimos a continuación:

  - *Ámbito total de la zona volada*: Partiendo del fichero vectorial de la zona volada se disuelve para obtener un único polígono con el área de la zona volada.
  - *Líneas de vuelo*: Partiendo del fichero vectorial de la zona volada se disuelve por el campo que almacena el código de línea de vuelo para obtener un único polígono por cada línea de vuelo, donde se almacena su código, el número de puntos contenidos en dicha línea de vuelo, el área del polígono y la densidad media de puntos en la línea de vuelo.
  - *Zonas de solape*: Partiendo del fichero de líneas de vuelo, obtenemos un nuevo fichero vectorial con los solapes entre pasadas y generamos una nueva capa vectorial de polígonos con los recubrimientos transversales entre pasadas. En la tabla adjunta se almacena el código de las dos líneas de vuelo que dan como resultado este solape, el porcentaje mínimo, máximo y medio de solape entre dichas pasadas.
  - *Agujeros*: Termina de generar este producto a partir del ámbito total de la zona volada.

- **analisispreviodatosdielmoopenlidar** Este algoritmo coge como entrada un listado de capas LiDAR y genera como salida un fichero vectorial de polígonos en formato shp que contiene un rectángulo con la extensión geométrica de cada uno de los ficheros LiDAR de entrada. Además, para cada polígono se añade una tabla adjunta con la información de la ruta del fichero de entrada, densidad aproximada de puntos por metro cuadrado, número total de puntos del fichero, área del rectángulo generado y alturas máxima y mínima dentro del fichero. Toda esta información se obtiene de la cabecera de los ficheros de entrada, por lo que se genera de forma muy rápida y sirve para tener una primera idea de si están todos los datos.
- **analisissolapedielmoopenlidar** Partiendo del fichero de líneas de vuelo, obtenemos un nuevo fichero vectorial con los solapes entre pasadas y generamos una nueva capa vectorial de polígonos con los recubrimientos transversales entre pasadas. En la tabla adjunta se almacena el código de las dos líneas de vuelo que dan como resultado este solape, el porcentaje mínimo, máximo y medio de solape entre dichas pasadas
- **precisionalturadielmoopenlidar** Este algoritmo coge como entrada un listado de capas LiDAR, una capa vectorial en formato shp de puntos con las zonas de control, un desplegable para elegir el campo que tiene la información de la altura en el shp de puntos de entrada y un parámetro numérico con el radio (en metros) con el que vamos a buscar. Para cada punto de la capa de entrada busca entre todas las capas LiDAR los puntos que caen dentro del radio indicado como entrada (suponemos que los puntos de control se han tomado en zonas planas) y realiza un cálculo estadístico con el error medio y desviación estándar, comparando la altura del punto de control con los puntos LiDAR encontrados dentro del radio indicado. Posteriormente, hacemos una segunda selección de puntos, eliminando los extremos que están fuera de una desviación estándar de la media (para eliminar puntos altos o bajos que nos añadan un error a la medida). Internamente se llama al algoritmo de agrupar puntos, para agrupar aquellos puntos de control que pertenecen a las misma zona de control, de forma que se puedan hacer informes de precisión en altura por cada punto, por grupo o

por el total de los puntos, por lo que también se añade un campo con la identificación de grupo en el SHP que se genera

CAPÍTULO 8

Pruebas, resultados y rendimiento

8.1 Introducción

Las pruebas de software, o testing consiste en los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador o videojuego. Es una fase en el desarrollo de software consistente en probar las aplicaciones construidas.

Las pruebas de software se integran en las diferentes fases del ciclo del software en la Ingeniería de software. De este modo se ejecuta un programa y mediante técnicas experimentales se trata de descubrir errores que pueda tener.

Para la determinación del nivel de calidad se deben efectuar una serie de medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones establecidas inicialmente del sistema.

Edsger Dijkstra dijo: "*El testing puede probar la presencia de errores pero no la ausencia de ellos*"

Para verificar productos complejos, de forma efectiva, el proceso de pruebas requiere de un proceso de investigación, más que seguir un procedimiento al pie de la letra. Una definición de "testing" es: *proceso de evaluación de un producto desde un punto de vista crítico, donde el "tester" (persona que realiza las pruebas) somete el producto a una serie de acciones inquisitivas, y el producto responde con su comportamiento como reacción.* Nunca se debe testear el software en un entorno de producción. Es necesario testear los nuevos programas en un entorno de pruebas separado físicamente del de producción. Para crear un entorno de pruebas en una máquina independiente de la máquina de producción es necesario crear las mismas condiciones que en la máquina de producción. Existen a tal efecto varias herramientas vendidas por los mismos fabricantes de hardware. Esas utilidades reproducen automáticamente las bases de datos para simular un entorno de producción.

Los informáticos distinguen, en general, entre errores de programación "bugsz defectos de forma. En un defecto de forma, el programa no realiza lo que el usuario espera. Por el contrario, un error de programación puede describirse como un fallo en la semántica de un programa. Éste podría presentarse, o no, como un defecto de forma si se diesen ciertas condiciones de cálculo.

Una práctica habitual es que el proceso de pruebas de un programa sea realizado por un grupo independiente de "testers" al terminar su desarrollo y antes de sacarlo al mercado. Una práctica que viene siendo muy popular es distribuir de forma gratuita una versión no final del producto para que los propios consumidores sean los que la prueben. En ambos casos, a la versión del producto en pruebas y que es anterior a la versión final (o "master") se denomina beta, y a dicha fase de pruebas, beta testing.

Puede además existir una versión anterior en el proceso de desarrollo llamada alpha, en la que el programa, aunque incompleto, dispone de funcionalidad básica y puede ser testeado.

Es cada vez más habitual que se realice una fase de RTM (Release To Market) testing antes de salir al mercado , dónde se comprueba cada funcionalidad del programa completo en entornos de producción.

Otra práctica es que el proceso de pruebas se realice desde que empieza el desarrollo y continúe hasta que finaliza.

En la cadena de valor del desarrollo de un software específico, el proceso de prueba es clave a la hora de detectar errores o fallas. Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado. Las aplicaciones de software han crecido en complejidad y tamaño, y por consiguiente también en costos. Hoy en día es crucial verificar y evaluar la calidad de lo construido de modo de minimizar el costo de su reparación. Mientras antes se detecte una falla, más barata es su corrección.

El proceso de prueba es un proceso técnico especializado de investigación que requiere de profesionales altamente capacitados en lenguajes de desarrollo, métodos y técnicas de pruebas y herramientas especializadas. El conocimiento que debe manejar un ingeniero de prueba es muchas veces superior al del desarrollador de software.

8.2 Pruebas de rendimiento

Las herramientas incluidas en el plugin han sido testeadas desde el principio de la implementación. Se han superado fallos por un mal entendimiento del problema, se han mejorado estrategias poco eficientes de búsquedas de puntos en herramientas como el visualizador de perfiles longitudinales. También se han buscado fórmulas para una correcta gestón de largos procesos de cálculo, como en el caso de los convertidores de formatos .LAS a .XYZ y de .XYZ a .LAS. En este caso ha sido de vital importancia la utilización de una tarea en segundo plano para ejecutarlos. De no ser así el hilo principal del programa habría quedado bloqueado no produciéndose ninguna respuesta hasta la finalización del proceso.

Se ha sometido a una simple prueba de stress a la gráfica longitudinal

obteniendo los siguientes resultados. Para la obtención de los tiempos de ejecución se han empleado código como este

```
...
long t1 = System.currentTimeMillis();
//obtengo la IGeometry de la franja de seleccion y creo el objeto TransformaPuntos3D para
    trasladar y rotar puntos

//arraylist de todas las capas lidar visibles en la vista
ArrayList<FLyrVect> capasLidarVisibles = obtenerCapasLidarVisibles();
ArrayList<double[]> todosLosRows=new ArrayList<double[]>();
int total=0;
for(int i=0;i<capasLidarVisibles.size();i++){

   FLyrVect lyrVect= capasLidarVisibles.get(i);
      if(LiDARDriver.isLidarLayer(lyrVect)){
         ArrayList<double[]> dat = getArrayPuntosFromGeom(geometry,lyrVect, i);
         todosLosRows.addAll(dat);
         total+= dat.size();
         }
      }

      log.info(new String("Se han obtenido "+total+ "puntos para el perfil en "+t4+"= ms.
          "));
      this.trans3d.setPuntos(todosLosRows);
      float[][] data=transformaPuntos(todosLosRows);
      capaSeleccionXZ = crearCapaAuxiliar(todosLosRows);
      ScatterPlotDialog panel2= new ScatterPlotDialog(565,400,data, trans3d, grosor,
          capaSeleccionXZ);
      long t2 = System.currentTimeMillis();
      long t3 = t2-t1;
      log.info("Tarda "+t3+"= ms. en mostrar la grafica");
```

Puntos leidos	Tiempo obtener puntos (ms)
616	248
98688	2196
178107	10983
1004241	48860
2004241	98860
616	248

Tabla 8.1: Tabla de tiempos de respuesta de la gráfica.

La prueba de stress ha puesto de manifiesto que se debería limitar de alguna manera el número de puntos a poder visualizar. Porque se obtienen tiempos de respuesta no tolerables. Otra posible estrategia podría consistir en lanzar la tarea de cálculo de la gráfica, en segundo plano, para no bloquear el hilo principal de ejecución de la aplicación.

En la herramienta de conversión de ficheros .LAS a .XYZ se han

obtenido los siguientes resultados:

Puntos .LAS	Tiempo convertirlos a .XYZ (ms)
22330853	24min40seg
42330853	44min10seg
92330853	127min24seg

Tabla 8.2: Tabla de tiempos de respuesta de la gráfica.

Como la tabla pone de manifiesto, resulta un proceso costoso la escritura en ficheros de texto de los datos de los puntos LiDAR. En ocasiones esta herramienta se ha dejado convirtiendo ficheros tras concluir la jornada laboral para poder disponer de los datos al día siguiente.

CAPÍTULO 9

Conclusiones

9.1 Síntesis del trabajo

En este proyecto final de carrera se han logrado los objetivos marcados: se ha dasarrollado un software para la gestión de un proyecto de datos LiDAR en el sistema de información geográfica gvSIG. En algunos aspectos se han logrado alcanzar metas y objetivos por encima de los planificados como es el caso de las herramientas avanzadas de conversión de formatos LAS a XYZ y viceversa.

Para un desarrollo de este tipo se requiere un análisis exhaustivo previo de gvSIG. Recabar información sobre SIG y gvSIG. Obtener y montar en Eclipse el código mínimo necesario para arrancar la aplicación. Incorporar las extensiones oficiales que se consideren interesantes añadiéndolas al workspace de Eclipse para adquirir funcionalidad complementaria. Obtener la documentación necesaria para el desarrollo de plugins y extensiones sobre gvSIG. Documentarse sobre el desarrollo de nuevos tipos de documentos sobre gvSIG. Conocer los elementos de la interfaz de éste SIG. Participar de herramientas de colaboración como la lista de correo de desarrolladores de gvSIG y también de SEXTANTE. La asistencia a eventos tales como las

Jornadas Internacionales de gvSIG y a las ponencias y talleres allí impartidos. Éstos puntos enumerados han sido necesarios para alcanzar objetivos e incluso superarlos en algunos aspectos.

Han aparecido problemas de distinta naturaleza durante el desarrollo del proyecto que se han conseguido solucionar, o bien se han buscado alternativas. Como en cualquier disciplina de ingeniería se ha de estar preparado para la resolución de problemas que puedan surgir. De esta forma desarrollar una experiencia y habilidad en la resolución de problemas de cualquier tipo. El conocimiento adquirido a través de personas más experimentadas en el dominio del problema es fundamental para este fin.

Pese a las dificultades que se hayan podido encontrar en el desarrollo de este proyecto, he obtenido, a nivel personal, una gran satisfación por su realización. La participación en un proyecto real, en una empresa como Dielmo 3D S.L. también ha sido una experiencia gratificante y enriquecedora tanto a título personal como profesional. Tras el paso por Dielmo 3D he podido incorporarme sin muchas dificultades al mercado laboral tratando temas de este ámbito. En la actualidad me encuentro realizando análisis y desarrollos de aplicaciones SIG basadas en gvSIG.

Entrando en más detalle se han utilizado herramientas como: *Eclipse* para la escritura, compilación y depuración del código Java del proyecto, el editor de imágenes *Gimp*, *UMLet* para la creación de diagramas UML, *GanttProject* para la elaboración del diagrama de Gantt de la planificación temporal, *Latex* y *Textmaker* para la realización de esta memoria, entre las más destacadas. En la elección de estas herramientas se ha seguido la filosofía de valorar y dar preferencia al software libre frente a software privativo.

Con los resultados obtenidos puede considerarse una herramienta muy útil tanto para la empresa Dielmo 3D como para cualquier miembro de la comunidad de usuarios de datos LiDAR. Resultó gratificante la visita a la empresa Dielmo 3D tras el término de las prácticas y observar como los técnicos allí presentes trabajan con la herramienta que

se ha desarrollado. Del mismo modo en la asistencia a las jornadas de gvSIG se pudo presenciar el interés despertado por el proyecto.

El desarrollo bajo la licencia GNU GPL aporta un potencial añadido a este proyecto. La comunidad de usuarios y desarrolladores pueden participar activamente en la creación de nuevas herramientas e incorporarlas al proyecto. De este modo en un futuro cercano el software estará dotado de la capacidad de resolución y análisis de problemas concretos. Podrá disponer de herramientas de análisis medioambiental, creación de modelos urbanos en 3D, sistemas de apoyo para planificaciones urbanísticas, infraestructuras públicas, etc., mantenimiento y gestión del tendido eléctrico, basados en datos LiDAR.

9.2 Colaboraciones

DielmoOpenLiDAR nace con el objetivo de ser una referencia a nivel mundial para el manejo de datos LiDAR con software libre, pero para conseguir este objetivo necesitamos todo el apoyo para continuar creciendo.

La idea de Dielmo es permitir a los usuarios añadir nuevos algoritmos que complementen los ya existentes o añadan nuevas aplicaciones, integrándolas de forma más sencilla y transparente, sin necesidad de que tengan que preocuparse de la parte gráfica, sino que únicamente implementen su algoritmo. Considerar que el uso de SEXTANTE es adecuado para esto y hará crecer el proyecto con las aportaciones de diferentes grupos de investigación interesados en desarrollar este tipo de herramientas.

Las colaboraciones pueden llegar de diferentes formas y diferentes grados de implicación. A continuación se exponen las principales formas de participación en el proyecto:

- **Dándole difusión**: Seguro que hay un gran número de usuarios en todo el mundo que no son capaces de manejar grandes volumenes de datos LiDAR y que están deseando conocer esta herramienta gratuita. Ayudar a darle difusión comentandolo con amigos, compañeros y colegas o poniendo un link a la web de

Dielmo http://www.dielmo.com/ .

- **Traduciendo**: Si se quiere colaborar con la traducción de DielmoOpenLiDAR y su documentación a otros idiomas, poniéndose en contacto con Dielmo para coordinar el trabajo.

- **Testeando**: Es costoso probar todas las combinaciones de situaciones de uso de un software. Precisamente uno de los puntos más fuertes del software libre es que el testeo se realiza de forma comunitaria. La idea de Dielmo es publicar constantemente versiones inestables de la aplicación, y sin los informes de error de los usuarios, se tardaría mucho más en ofrecer un producto estable y maduro. Por esta razón una de las formas de participación más apreciadas es el testeo del programa.

- **En las listas de correo**: Las listas de correo son el centro de la comunidad gvSIG, y una de las piezas más importantes a través de las cuales se canalizan la mayoría de las comunicaciones entre usuarios, desarrolladores y gestores del proyecto. Participando en las listas de correo se fomenta la comunidad, se resuelven dudas técnicas, se discuten mejoras, etc. Tanto usuarios como desarrolladores, ante cualquier problema, funcionamiento inesperado o simplemente duda, acudir a las listas de correo para exponerlas porque es posible haber encontrado un error en el programa y la aportación ayudará a que en la próxima versión de gvSIG o de DielmoOpenLiDAR éste haya sido resuelto.

- **Escribiendo**: Si se realizan tutoriales o artículos en blogs, revistas o cualquier otro medio de comunicación , relacionados con el uso de DielmoOpenLiDAR, Dielmo puede ofrecer enlaces a estos contenidos a través de su web para que el resto de la comunidad pueda acceder a ellos de forma sencilla.

- **Desarrollando**: Si se trabaja en el desarrollo de algoritmos para el tratamiento de datos LiDAR, es posible aprovechar DielmoOpenLiDAR como base para los desarrollos, facilitando la implementación de los algoritmos. Dielmo estará encantada de incluir extensiones dentro de la aplicación, de forma que otros usuarios puedan aprovecharse del desarrollo, a la vez que lo da a conocer a una gran cantidad de usuarios LiDAR. Por otro lado, Dielmo está abierta a trabajar

en desarrollos conjuntos con otros grupos de investigación para el desarrollo de módulos orientados a cualquier aplicación de los datos LiDAR.

- **Aportando algoritmos** : Si se trabaja en el desarrollo de algoritmos para el tratamiento de datos LiDAR pero por cualquier motivo se prefiere trabajar en otro lenguaje de programación o sobre otra plataforma, también es posible la colaboración con Dielmo aportando algoritmos o código fuente para que Dielmo los implemente dentro de DielmoOpenLiDAR haciendo referencia al autor de los algoritmos.
- **Patrocinando**: Para el desarrollo de este software libre, DIELMO asume una serie de riesgos financieros que en parte espera cubrir a través de la colaboración económica de empresas e instituciones en forma de patrocinio o financiando el desarrollo en más profundidad de herramientas para aplicaciones específicas como el uso del LiDAR para inventarios forestales, etc.

9.3 Perspectivas de futuro y mejoras.

Con las conclusiones obtenidas en el punto anterior el futuro inmediato del proyecto pasa por una capacidad de crecimiento del mismo gracias a la licencia GNU GPL. Esta licencia persigue garantizar la libertad para compartir y modificar todas las versiones de un programa y asegurar que permanecerá como software libre para todos sus usuarios. Licencias Públicas Generales están destinadas a garantizar la libertad de distribuir copias de software libre y cobrar por ello si quiere, a recibir el código fuente o poder conseguirlo si así lo desea, a modificar el software o usar parte del mismo en nuevos programas libres, y a hacer saber que puede hacer estas cosas.

Actualmente hay una tendencia a que estén disponibles cada vez más datos LiDAR que cubren grandes extensiones del territorio y de aquí a unos años toda esta información será libre para que cualquiera pueda acceder a ella. El Plan Nacional de Ortofotografía Aérea, PNOA, proporcionará datos LiDAR gratuitos de toda España. En la figura 9.1 se muestra el plan de vuelo para el 2009 que incluye la adquisición de datos LiDAR.

Figura 9.1: Plan de vuelo PNOA 2009.

Dielmo ha desarrollado un servidor de datos LiDAR que permite la visualización de datos lidar alojados en un servidor. Para la realización del cliente web que permite visualizar los datos se ha adaptado una herramienta procedente de este proyecto. Se trata de la herramienta de visualización de perfiles longitudinales que ha sido adaptada y mejorada. Es una satisfacción personal que el trabajo realizado sea útil para la realización de otros proyectos como en este caso.

Dielmo tiene previsto permitir la creación de flujos de trabajo de forma gráfica a partir de las funciones existentes, de forma que se puedan automatizar los procesos siguiendo el orden y algoritmos definidos por el usuario y ejecutarlos por lotes. Por muy sofisticados que sean los métodos de clasificación automática, es imposible el tener un 100 % de aciertos, por lo que para que este software sea útil no solamente para investigación sino también para producción, se hace necesaria la implementación de herramientas que permitan realizar un adecuado control de calidad de los productos finales descritos en los puntos anteriores con herramientas de edición manual inteligente, semiautomática, para realizar las correcciones necesarias de la forma más rápida y precisa posible. Las siguientes fases del proyecto consisten en desarrollar herramientas inteligentes para la creación de nuevos productos finales de valor añadido como:

- Vectorización de edificios y estructuras.

- Vectorización de líneas eléctricas y localización de objetos, como árboles o construcciones ilegales, que se acercan a la línea eléctrica para su mantenimiento.
- Estimación de parámetros forestales como contar árboles, estimar la altura de los árboles, medir la altura media de un rodal, obtener parámetros de calidad de la madera, etc.
- Extracción de curvas de nivel para la actualización de cartografía digital.
- Detección de cambios para localizar construcciones ilegales, evolución de la línea de costa, etc.
- Mejora de los MDT para su utilización en estudios hidráulicos.

CAPÍTULO 10

Anexos

10.1 Manual de usuario DielmoOpenLidar

DielmoOpenLiDAR 2.0
para gvSIG 1.1.2

Manual de usuario

Fecha: **14/1/2010**

Índice de contenido

1.- Introducción....3
2.- Driver para el acceso a datos LiDAR en gvSIG....4
3.- Configuración de las preferencias....5
4.- Manejo de datos LiDAR en gvSIG....6
5.- Herramientas para la gestión de proyectos (el documento LiDAR)....13
6.- Más información, actualizaciones, actualizaciones y datos de muestra 28

1.- Introducción

Después de la adquisición de datos LiDAR, disponemos de una gran nube irregular de puntos XYZ. Imaginemos al avión volando durante más de cuatro horas al día, midiendo 150.000 puntos por segundo, y para cada uno de esos puntos se guardan las coordenadas XYZ del primer y último pulso, el valor de la intensidad de la señal que vuelve hasta el sensor, el tiempo GPS, etc.

Hace unos años había muy pocos datos LiDAR disponibles, limitándose a pequeños proyectos para zonas inundables, etc., pero gracias al Plan Nacional de Ortofotografía Aérea (PNOA) habrá disponibles datos LiDAR de toda España actualizados cada 4 años y toda esta información será libre para que cualquiera pueda acceder a ella.

La intención de este software es ir más allá de la visualización de los datos como se consiguió en la primera versión del software DielmoOpenLiDAR 1.0., es decir, usar el driver realizado para trabajar en aplicaciones que realmente consiguen otros productos de salida como el control de calidad de la ejecución vuelos LiDAR financiada por el Instituto Geográfico Nacional de España.

Para ello DIELMO 3D S.L. aporta su experiencia investigadora en el ámbito de la teledetección, el desarrollo de software para la generación y tratamiento de modelos digitales del terreno y para el procesado de datos LiDAR, además del conocimiento del mercado al ser uno de los principales proveedores de datos LiDAR en España y haber trabajado en numerosos proyectos de adquisición y procesado de datos para diferentes aplicaciones.

A continuación se hace una descripción de la metodología y las herramientas que hemos planteado en el desarrollo de este software libre llamado DielmoOpenLiDAR 2.0.

2.- Driver para el acceso a datos LiDAR en gvSIG

En 2008 DIELMO comenzó el desarrollo del software libre DielmoOpenLiDAR 1.0 con una fase inicial que consiste en un driver para el acceso a datos LiDAR en gvSIG. Este driver es la base para el manejo de los datos en diferentes formatos estándar, de forma que se ha dotado a gvSIG con las herramientas básicas que permiten a los desarrolladores el trabajar con datos LIDAR de la forma más transparente y sencilla posible. Por otro lado, también se permite abrir datos LiDAR originales (LAS y BIN), visualizarlos superpuestos a cualquier otra información geográfica, consultar los valores originales de cada uno de los puntos y editarlos, realizar un análisis visual de los datos y un control de calidad de los mismos.

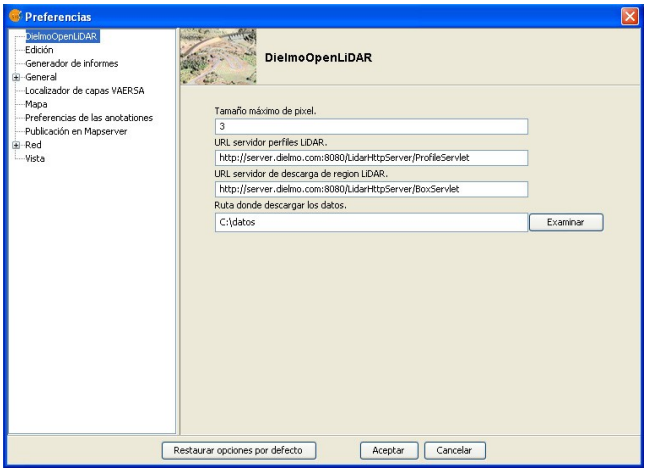
DielmoOpenLiDAR está pensado para manejar grandes volúmenes de datos LiDAR (cientos de GigaBytes) de una forma rápida y robusta. Para ello, no carga los datos en memoria y solamente pinta los datos necesarios en función de la escala de visualización.

En concreto, hemos añadido a gvSIG las siguientes funcionalidades:

- Driver para el acceso a los datos LiDAR en formato LAS y BIN para lectura y escritura.
- Botones para aplicar simbología automática en función de la altura, intensidad, clasificación o color a todas las capas LiDAR disponibles en una vista de gvSIG.
- Botón para volver a aplicar una leyenda por defecto a las capas LiDAR, de forma que cada una se pinte con un símbolo único aleatorio.

3.- Configuración de las preferencias

Para entrar en las preferencias hay que ir al menú ventana/preferencias o pulsar el botón correspondiente en la vista y luego seleccionar el apartado DielmoOpenLiDAR y nos aparecerá esta ventana:

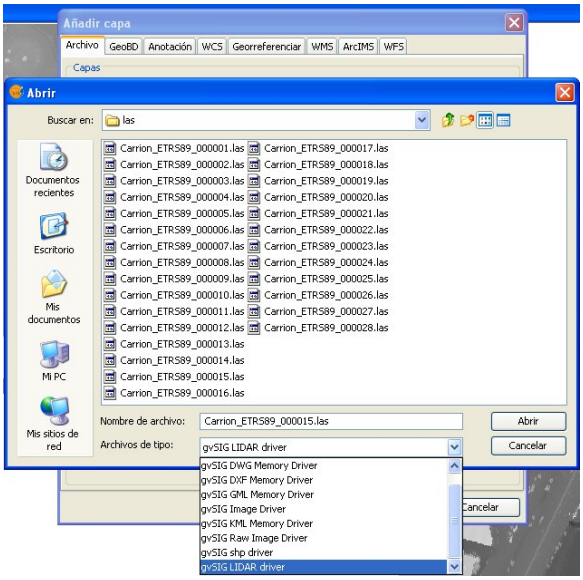


A la hora de visualizar los datos LiDAR en la vista, dependiendo de la escala de visualización los puntos se pintan con tamaños diferentes. Por ejemplo, cuando estamos muy lejos se pinta solamente un pixel por punto y cuando estamos muy cerca se pintan los puntos con un grosor mayor para que se visualicen mejor. El primer parámetro es el tamaño máximo del pixel que indica el grosor máximo que tendrán los puntos. Este parámetro es interesante a la hora de superponer los datos LiDAR con ortofotos, de forma que podamos ajustar el grosor para una mejor visualización.

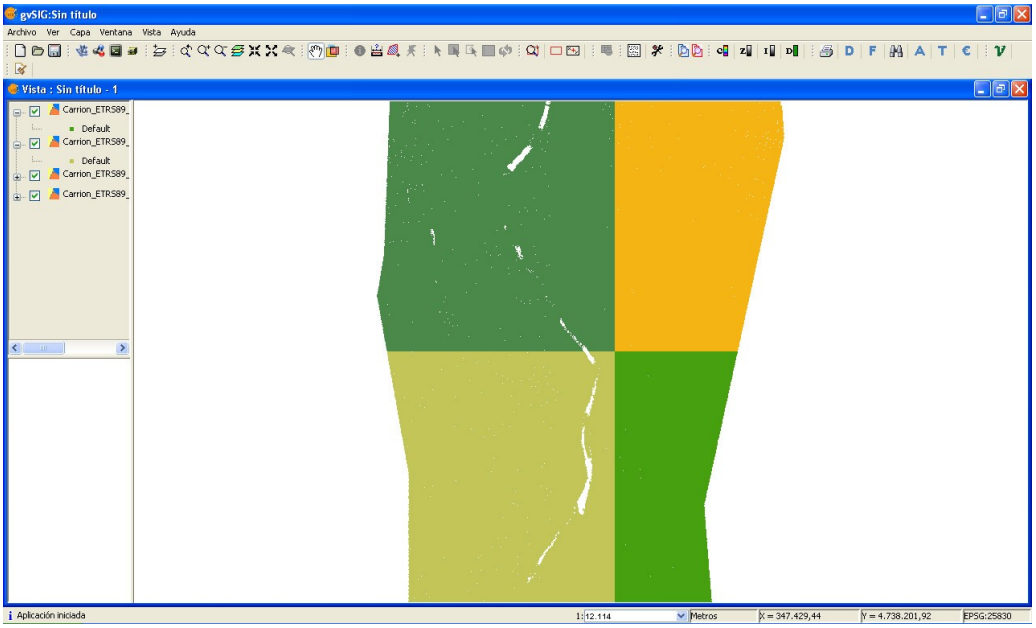
Por otro lado tenemos dos parámetros para configurar a donde hacer las peticiones de perfiles y descargas por región LiDAR. De manera que si conocemos de algún servicio que de esta posibilidad nosotros podríamos hacerle peticiones configurando estos parámetros. Además, si quisiéramos guardarnos en una ruta los datos que descargamos en las peticiones de descarga por región, tenemos un cuarto parámetro que indica la ruta donde descargar los datos. Podemos hacer uso de estas descargas una vez configurado todo desde el menú LiDAR->Descargar datos LiDAR y desde ahí elegir entre descargar perfiles o descargar por región.

4.- Manejo de datos LiDAR en gvSIG

Una vez instalada la extensión aparecerá un nuevo driver (gvSIG LiDAR driver) en la ventana de añadir capa por fichero, y es el que debemos seleccionar para abrir un fichero LiDAR en formato LAS o BIN.

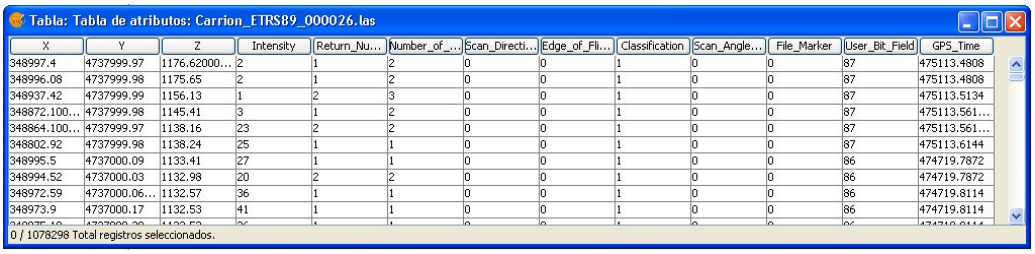


Al abrir un fichero LiDAR, éste se añade al TOC de la vista como si se tratara de una capa vectorial de puntos, aplicando un color aleatorio para cada capa. A continuación vemos como aparecen 4 ficheros LiDAR por defecto al abrirlos en gvSIG.



El driver para la visualización de los datos LiDAR en gvSIG tiene una estrategia que permite dibujar, seleccionar e imprimir los millones de puntos que pueden existir en un fichero LiDAR. La estrategia contemplada sigue la idea de que según la resolución a la que el usuario este visualizando su capa LiDAR se dibujaran más o menos puntos, de forma que se aligeren las tareas de representación, teniendo en cuenta que se quieren manejar miles de millones de puntos a la vez. Por este motivo, hay que tener en cuenta que cuando el zoom está alejado es posible que no se estén pintando todos los puntos del fichero, y se puede considerar como una previsualización hasta que no nos acercamos los suficiente.

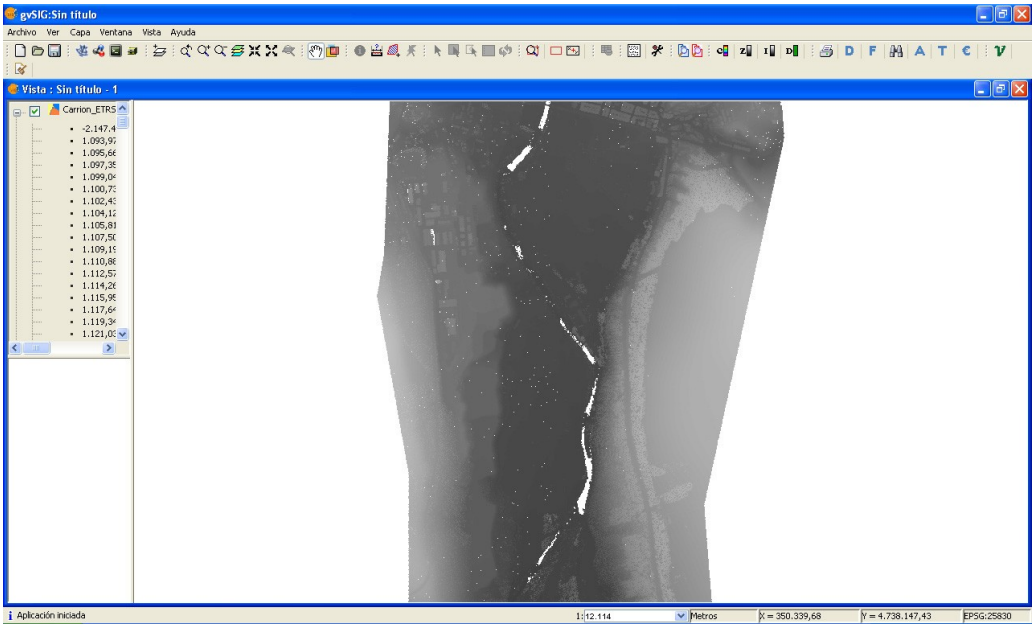
Estos ficheros tienen una tabla asociada que en función del formato tendrá más o menos columnas y que contiene información asociada a cada punto como la altura, intensidad, clasificación, tiempo GPS, etc. Al comenzar la edición podemos cambiar los valores de esta tabla, pero no podremos modificar la estructura de la tabla al no permitirlo los formatos de este tipo de datos.



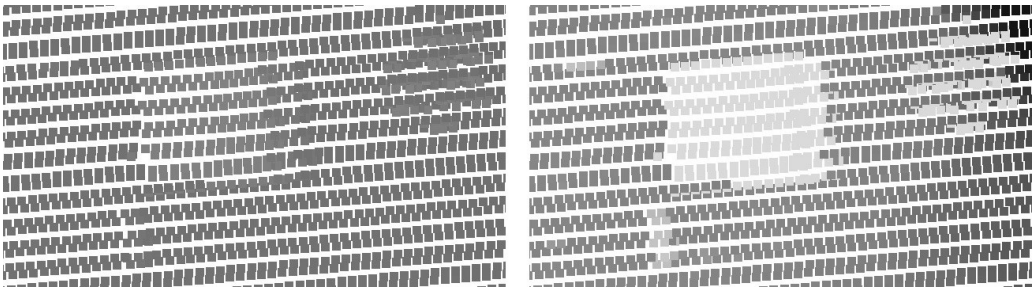
Al abrir estos ficheros en la vista aparece esta nueva barra de botones que utilizaremos para cambiar de forma automática la simbología de las capas LiDAR que tengamos abiertas en la vista.

A continuación pasamos a describir lo que hace cada uno de estos botones.

Pinta los datos LiDAR en función de la altura. Al pinchar en este botón se realiza un cálculo estadístico de los valores de altura de los puntos que caen dentro de la vista. De esta forma, podemos ir ajustando la simbología para que se visualicen con más detalle las zonas que nos interesan. Esta simbología se aplica a todas las capas LiDAR disponibles en la vista, de forma que los datos se visualizan todos con la misma simbología y no se nota donde está el cambio de una capa a otra. Por ejemplo, a continuación vemos los mismos datos que en la figura anterior, pero después de haber pulsado este botón:

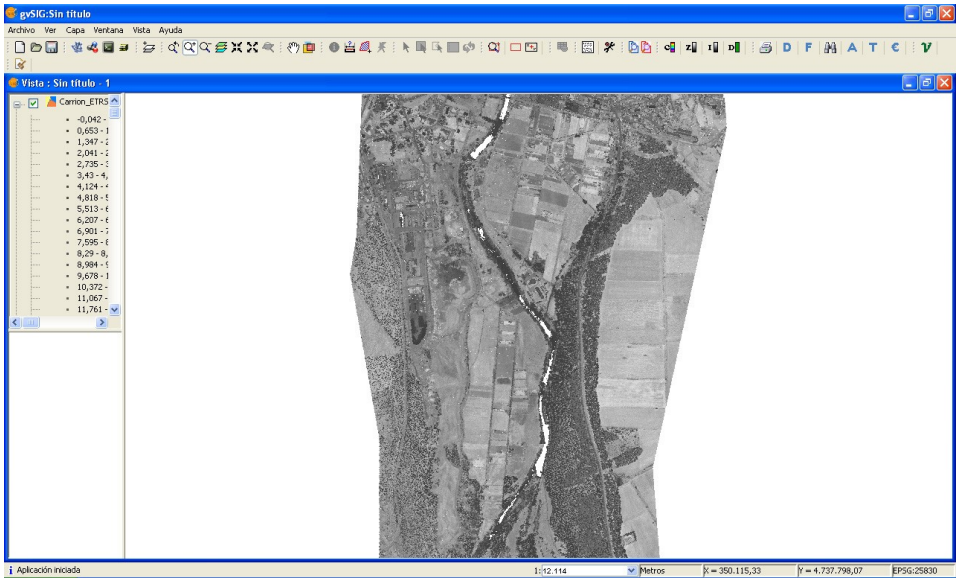


Al hacer zoom en una zona más pequeña, si tenemos ajustado el rango de colores para visualizar toda la extensión de los ficheros u otra zona de los mismos, es posible que no se visualizen los datos con mucho detalle (ejemplo a la izquierda en la figura siguiente), pero si volvemos a pulsar sobre este botón, se volverá a calcular la leyenda en función de los datos que estamos visualizando (ejemplo a la derecha en la figura siguiente), de forma ahora se puede distinguir mejor el rango de alturas que tenemos en pantalla.



Pinta los datos LiDAR en función de la intensidad. Al pinchar en este botón también se realiza un cálculo estadístico de los valores de intensidad de los puntos que caen dentro de la vista. De esta forma, podemos ir ajustando la simbología para que se visualicen con más detalle las zonas que nos interesan. Esta simbología se aplica a

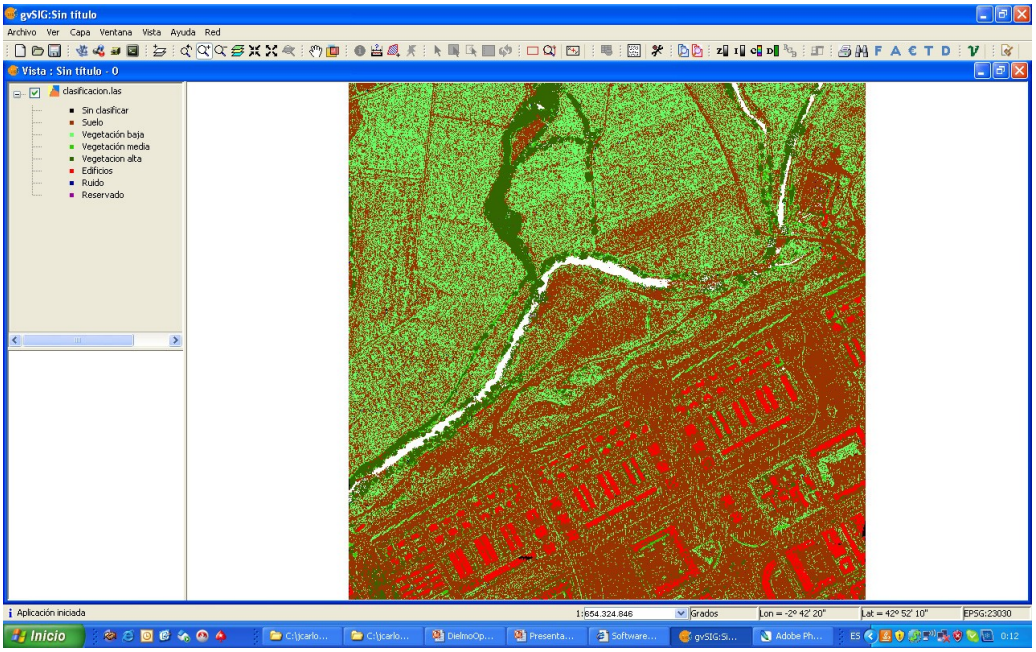
todas las capas LiDAR disponibles en la vista, de forma que los datos se visualizan todos con la misma simbología y no se nota donde está el cambio de una capa a otra. Por ejemplo, a continuación vemos los mismos datos que en los ejemplos anteriores pero pintados en función de la intensidad:



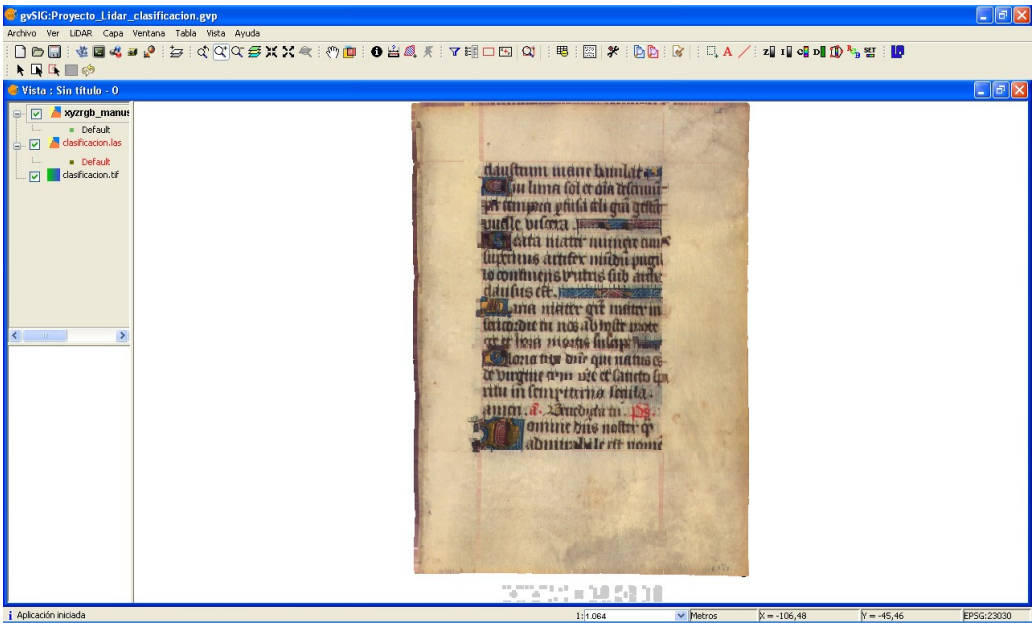
Al igual que en el caso de las alturas, si cada vez que pulsamos a este botón se calcula una nueva simbología en función de los valores de intensidad de los puntos que caen dentro de la vista, y esto se puede utilizar para buscar el rango que mejor se ajusta a nuestras necesidades.

 Pinta los datos LiDAR en función de los valores de la tabla de clasificación que aparece en el documento LiDAR. Por lo tanto, para que este botón surta efecto, previamente tendremos que:

- Crear un documento LiDAR.
- Definir la clasificación que deseamos usar dentro del documento LiDAR activo que queramos usar.



En capas que disponen de color rgb (formatos BIN y LAS 1.2) es posible pintarlas con su color correspondiente. Como puede haber capas en la vista que no dispongan de rgb, cuando se pulsa este botón a esas capas se le aplicara la leyenda por defecto. Sino hay capas que tengan rgb este botón quedara inhabilitado. En la figura siguiente vemos como queda una capa de datos pintada en función del RGB definido para cada punto (en este caso se trata de un LiDAR terrestre midiendo un manuscrito).

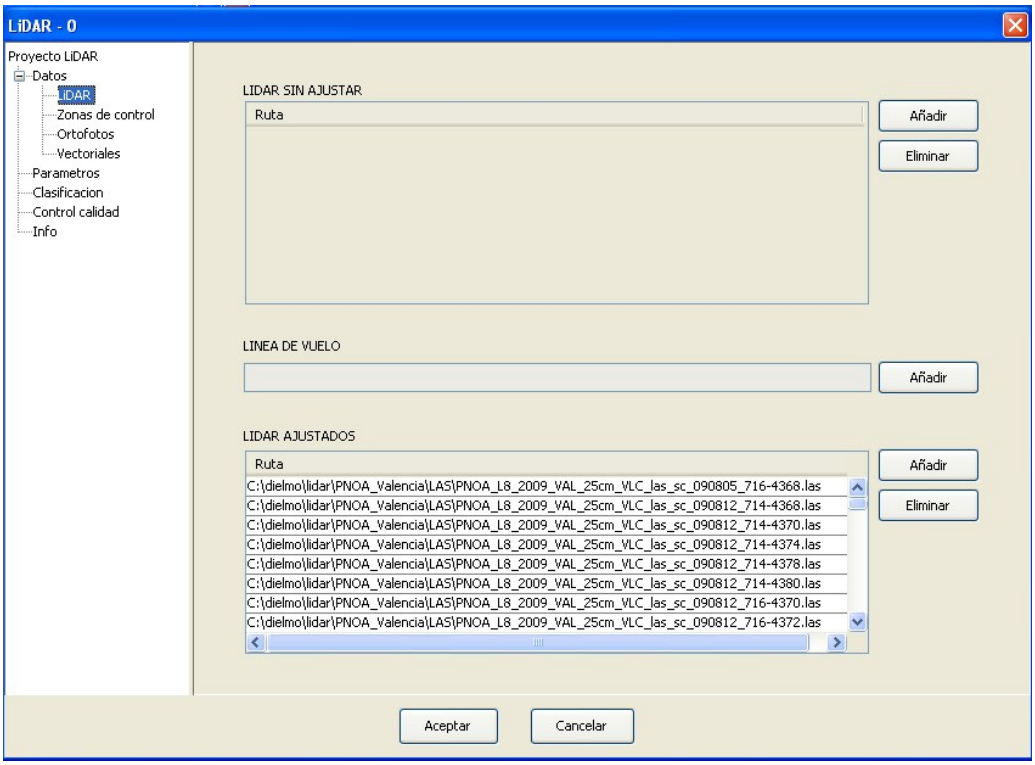


Vuelve a pintar los datos LiDAR con la simbología por defecto, aplicando un color aleatorio a cada una de las capas. De esta forma podemos saber la extensión que ocupa cada uno de los ficheros.

5.- Herramientas para la gestión de proyectos (el documento LiDAR)

Una vez establecidas en la fase anterior las bases para los desarrollos relacionados con datos LiDAR en gvSIG, la segunda fase consiste en desarrollar todas las herramientas necesarias para poder gestionar de forma más rápida y sencilla grandes volúmenes de datos LiDAR junto con otros datos GIS como ortofotos o cartografía vectorial. Para ello hemos creado un nuevo tipo de documento en gvSIG a parte de las vistas, tablas y mapas.

Dentro de este nuevo tipo de documento, definiremos todos los datos relacionados con el proyecto LiDAR, de forma que éstos se definirán una sola vez al principio y posteriormente desde la vista a la hora de cargar los datos o a la hora de lanzar los algoritmos de cálculo ya no será necesario indicar donde están los datos ni donde se deben de guardar los resultados, porque todo eso ya estará indicado en el documento LiDAR. A continuación vemos cada una de las ventanas que componen al documento LiDAR.

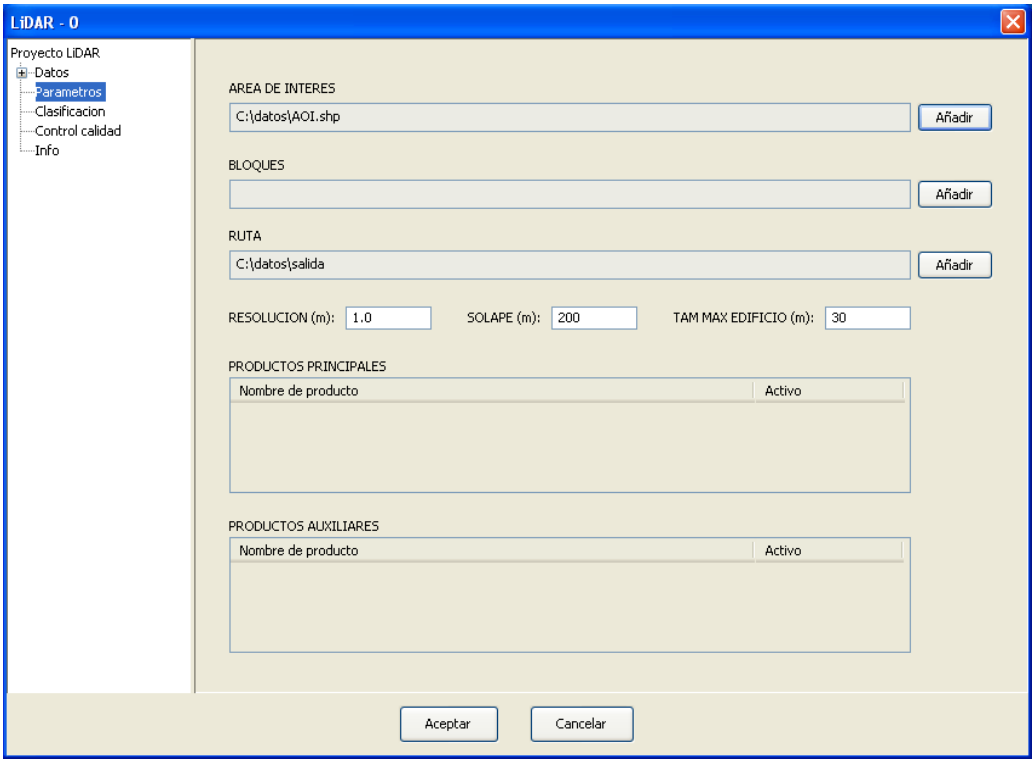


Dentro del documento LiDAR, a la izquierda tenemos un árbol que nos permite seleccionar entre varias ventanas para ir indicando todos los datos relacionados con un proyecto LiDAR. A continuación hacemos una descripción de la información que se introduce en cada una de las ventanas:

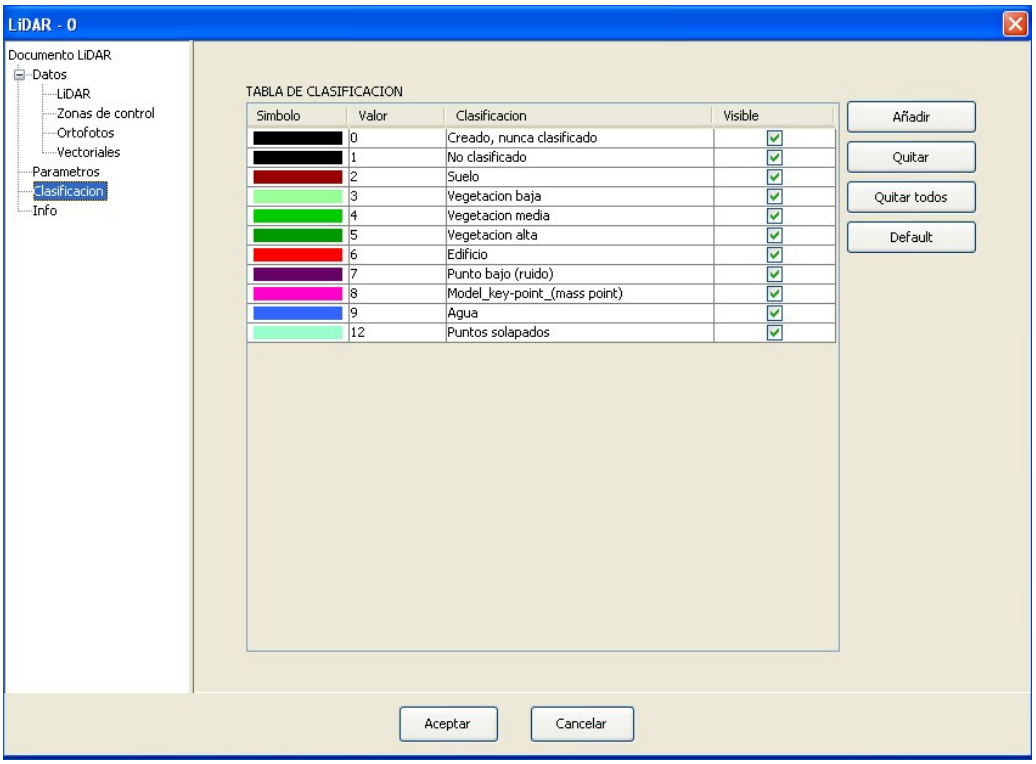
- **Datos LiDAR**: Corresponde con la figura anterior. Aquí indicaremos la ruta de todos los ficheros LiDAR de los que dispongamos para el proyecto en cuestión. Si a los datos LiDAR ya se les ha aplicado un ajuste de las variaciones de altura entre pasadas, los añadiremos dentro del apartado LiDAR ajustados. Por el momento no hay algoritmos que usen los LiDAR sin ajustar pero como previsión para el futuro se deja reservado un apartado para añadir los datos LiDAR sin ajustar y un vectorial con los polígonos de las líneas de vuelo.
- **Zonas de control**: Aquí se definen las rutas de los ficheros que contienen los levantamientos GPS que se tomarán como zonas de control.
- **Ortofotos**: Aquí se definen las rutas de las ortofotos de la zona, que podremos cargar de forma automática

desde la vista.

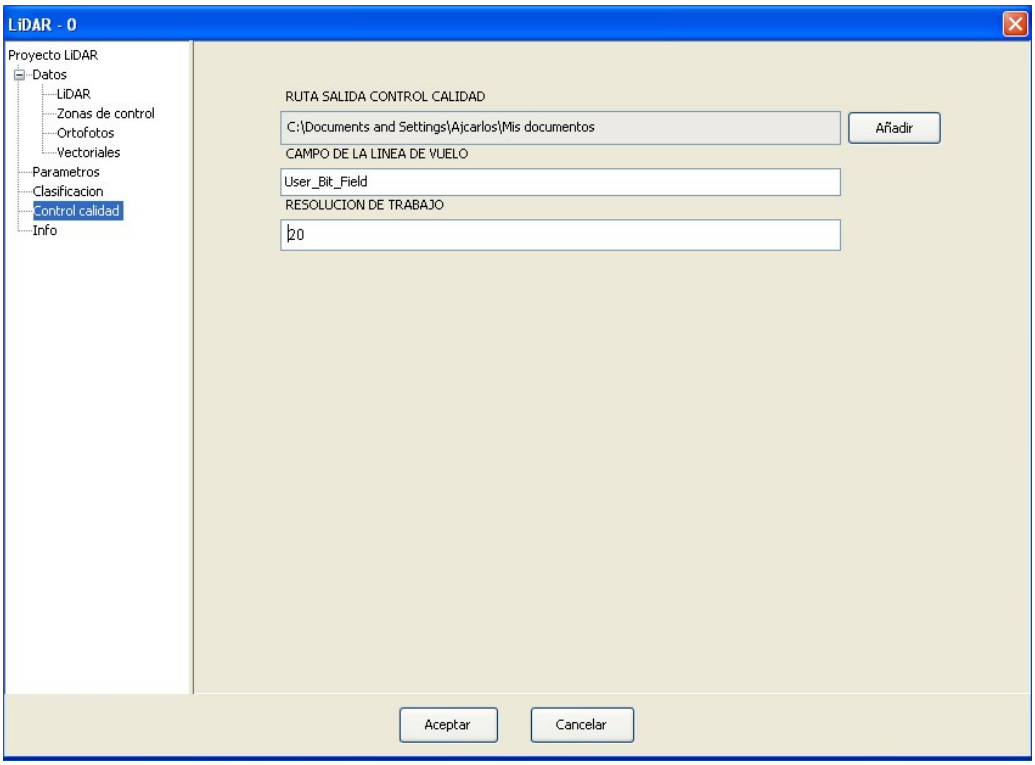
- **Vectoriales**: Aquí se definen las capas vectoriales (en cualquiera de los formatos soportados por gvSIG) asociados a la zona de estudio. También se puede definir una leyenda por defecto para cada una de las capas vectoriales, de forma que cuando se carguen automáticamente desde la vista lo hagan con dicha simbología.
- **Parámetros**: Corresponde con la imagen siguiente. Aquí se definen las rutas del área de interés (fichero shp de polígonos) que define la zona a procesar. Los bloques (fichero shp de polígonos) en los que vamos a cortar los datos para poder procesarlos de una forma más ágil (por ejemplo en bloques de 2x2 km). La ruta de salida donde se guardarán los resultados. La resolución con la que vamos a generar los productos de salida y el solape que se dará a cada uno de los bloques a la hora de procesar los datos para conseguir una correcta unión entre bloques después de hacer el mosaicado de los productos finales. Este apartado también esta reservado para futuros algoritmos y formas de trabajo, pero de momento se puede usar para definir el área de interés que nos facilita desde el cargador de capas el centrarnos la vista en nuestra zona de trabajo.



- **Clasificación**: Corresponde a la imagen de la figura siguiente. Aquí definiremos la tabla de clasificaciones que tendremos en cuenta a la hora de visualizar los datos clasificados y a la hora de interpretar los valores del campo Clasificación en la tabla correspondiente a cada uno de los puntos. Esto sustituye a las leyendas que se usaban en la fase 1 del desarrollo. Este panel nos permite cambiar el color con el que se pinta cada una de las clases, cambiar el valor con el que se clasifica cada una de las 11 clases por defecto (definidas como estándar en el formato Las 1.1 y superiores), añadir nuevas clases y activar las que serán visibles o invisibles en la vista.



- **Control de calidad**: Este panel sirve para indicar la ruta de salida de los ficheros vectoriales resultado del control de calidad, para indicar cual es el campo que contiene el código de línea de vuelo en los ficheros LiDAR y definir la resolución de trabajo para el análisis de la ejecución del vuelo (se describirá más adelante).



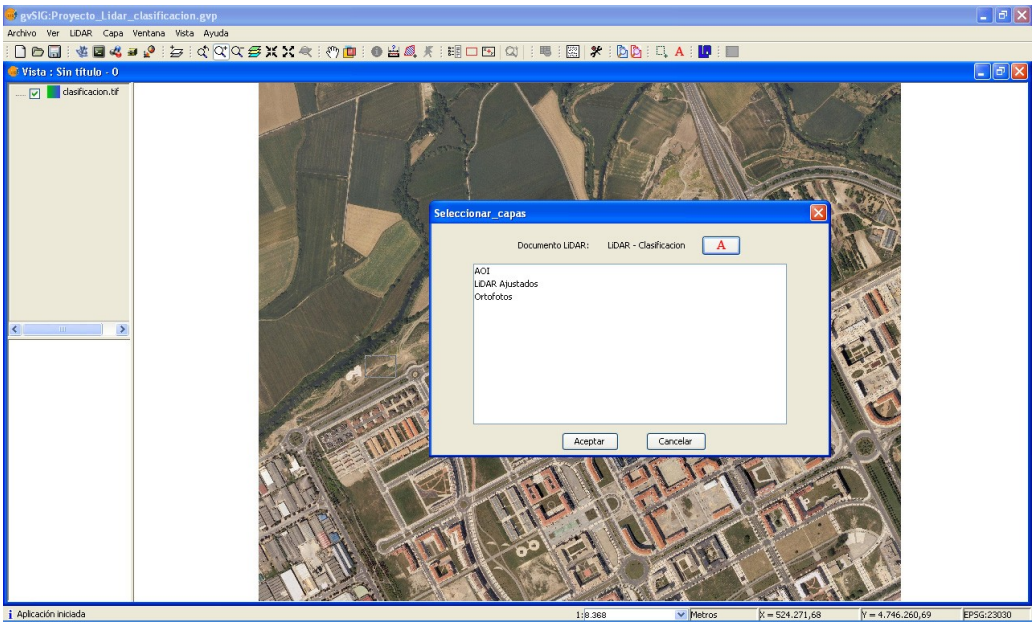
- **INFO**: Este panel se reserva para definir información adicional como el operador que está trabajando con los datos o la ruta del fichero log donde se irán guardando información de los procesos que se ejecuten.

Al pulsar el botón de Aceptar, se recopila la información de las rutas de los ficheros en diferentes formatos (LiDAR, vectoriales y raster) y la extensión geográfica que ocupa cada fichero de forma que esta información se pueda utilizar desde la vista cuando usemos la herramienta que carga capas de forma automática.

Una vez definidos los datos disponibles relacionados con el proyecto LiDAR, podemos volver a la vista y trabajar con los datos de una forma mucho más ágil. La nueva barra de herramientas LiDAR disponibles en la vista en esta segunda fase del proyecto es:

A continuación pasamos a describir los nuevos botones que no estaban en la fase 1.

Cargador de capas automático. Previamente, en la descripción de la fase 1 hemos visto como para cargar una capa teníamos que pulsar el botón de añadir capa y buscar la ruta del fichero que queremos abrir. Este proceso parece sencillo cuando estamos trabajando con poca información, pero no es trivial si tenemos en cuenta la cantidad de datos de la que vamos a disponer a nivel de toda España. En este caso, para encontrar entre todos esos ficheros el que cae dentro de la zona que queremos visualizar o aunque lo supiéramos, solamente el buscar el nombre del fichero entre los que tendremos en la misma carpeta sería un proceso muy lento. Por este motivo, la primera de las herramientas que se han añadido a la vista en esta fase 2 del proyecto consiste en este cargador de capas automático. Al pulsar este botón, se nos permite dibujar un rectángulo dentro de la vista y posteriormente se hace una búsqueda automática entre todos los datos geográficos definidos previamente en el documento LiDAR, mostrando esta ventana para elegir el tipo de dato que queremos cargar:



En la parte superior de la venta se indica el nombre del documento LiDAR activo, a partir del cual se cogerán los datos geográficos. Si quisiéramos cambiar el documento LiDAR activo, podemos pulsar el botón "A" que se describirá en el siguiente punto.

La ventana de selección de capas automática nos muestra un listado de todos los datos geográficos que están definidos en el documento LiDAR activo y que caen dentro del rectángulo que previamente hemos dibujado en la vista. Podemos seleccionar uno o varios elementos, y al pulsar el botón Aceptar estos datos se añaden a la tabla de contenidos de la vista activa y se visualizan superpuestos al resto de información que hubiera disponible.

Selección del documento LiDAR activo del cual se cogerán los datos de entrada/salida para cargar capas de forma automática o para lanzar los algoritmos de cálculo. Este modo de trabajar permite definir diferentes documentos LiDAR a la vez en un mismo proyecto de gvSIG y es muy útil en el caso de trabajar con una serie temporal de datos de la misma zona. Por ejemplo, en el caso de Gipuzkoa hay disponibles datos LiDAR del año 2005 y del año 2008, los ficheros correspondientes a cada uno de los años podrían estar definidos en un documento LiDAR diferente, y desde la vista podríamos seleccionar de qué año queremos cargar las capas de forma automática, de forma que podamos estudiar la evolución de la superficie terrestre. El documento LiDAR activo por defecto es el primero de la lista o el último que se ha utilizado. A continuación vemos la ventana que aparece al pulsar este botón para seleccionar el documento que queremos activar:

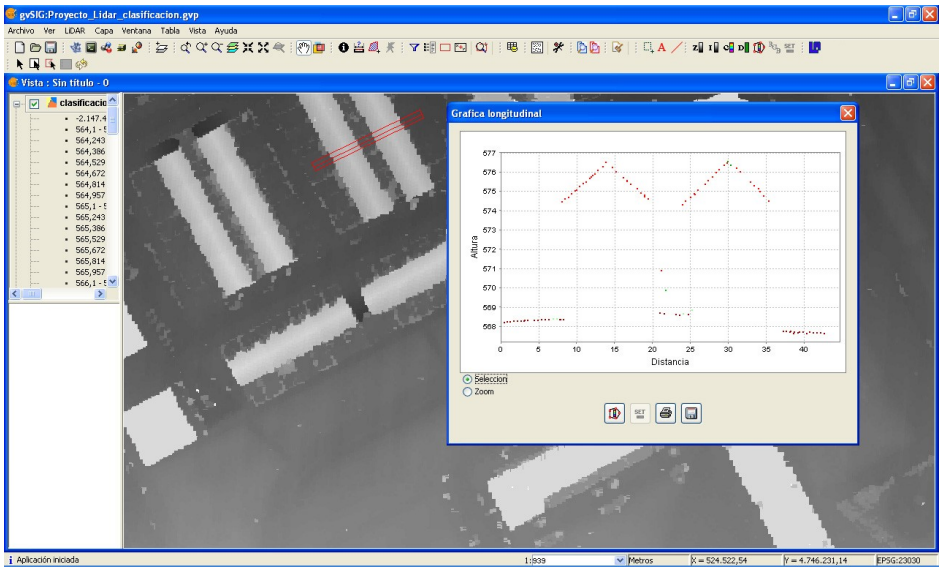
Selección por clasificación: Esta herramienta nos muestra una ventana con el listado de clases definido en el documento LiDAR activo y posteriormente parte de los puntos LiDAR que están seleccionados entre una o varias capas de la vista activa y realiza un filtro que restringe la selección solamente a los puntos que pertenecen a las clases seleccionadas.

Establecer clasificación: Esta herramienta nos muestra la misma ventana que en el caso anterior con el listado de clases definido en el documento LiDAR activo y posteriormente asigna la clasificación deseada a los puntos seleccionados entre una o varias capas de la vista activa. Esta herramienta solamente está activa cuando las capas LiDAR están en edición y cuando el nivel de zoom es lo suficientemente alto como para pintar todos los puntos en la vista.

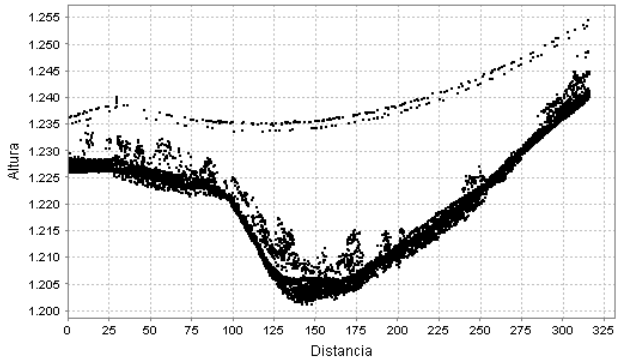
Perfil longitudinal. Esta herramienta permite un análisis más profundo de los datos así como hacer clasificaciones manuales o corregir las clasificaciones automáticas de los datos LiDAR, permitiendo realizar un adecuado control de calidad de los datos.

La herramienta se activa cuando hay alguna capa LiDAR en la vista y al pinchar este botón se nos permite definir la región de datos LiDAR para los cuales queremos visualizar el perfil longitudinal. Para ello en primer lugar definiremos una línea pinchando sobre dos puntos en la vista, y posteriormente alejando el ratón de la línea y volviendo a pinchar sobre la vista, podremos definir la profundidad del perfil.

Una vez definida la zona, aparece esta ventana con el perfil longitudinal de los puntos que hemos seleccionado. Por ejemplo, en la figura siguiente vemos un perfil longitudinal sobre dos edificios, y podemos identificar perfectamente los puntos que han caído en el tejado de los edificios, así como los puntos que son suelo y también observamos unos puntos que son vegetación en medio de los dos edificios.



Dentro de la gráfica, los puntos se pintan en función de su clasificación según la tabla de clasificaciones definida en el documento LiDAR activo. Si los puntos no estuvieran clasificados o no hubiera un documento LiDAR, estos se pintarán en el color que se haya definido para la clase creado, nunca clasificado. En la figura siguiente vemos otro ejemplo de gráfica longitudinal sobre una línea eléctrica. En este caso todos los puntos salen del mismo color porque el fichero las no está clasificado.



Sobre la gráfica solamente se pintan los puntos LiDAR que se estaban visualizando en la vista en el momento que se definió la región del perfil, por lo que hay que tener en cuenta dos cosas:

- Si se define la región del perfil mientras se están pintando puntos en la vista, en la gráfica longitudinal solamente aparecerán aquellos puntos que ya se habían pintado en la vista, y no los que se pinten posteriormente.
- Como ya comentamos previamente, la estrategia del driver para la visualización de los datos LiDAR en gvSIG pinta menos puntos en la vista conforme vamos alejando el nivel de zoom, por lo tanto si hacemos un perfil con un nivel de zoom alejado hay que tener en cuenta que no se están visualizando todos los puntos.

Dentro de la gráfica longitudinal tenemos una serie de herramientas que nos permiten analizar los datos de una forma más ágil. A continuación pasamos a describir cada una de estas herramientas.

- **Selección**: Permite definir sobre la gráfica longitudinal un polígono para seleccionar los puntos deseados. Una vez definido el polígono, los puntos seleccionados se pintan en color naranja dentro del gráfico longitudinal y los mismos puntos también quedan seleccionados en la vista.

- **Zoom**: Permite hacer un zoom dentro de la gráfica longitudinal para ver con más detalle los puntos deseados o seleccionar con más precisión. Si pulsamos sobre la gráfica con el botón derecho del ratón volvemos al zoom original.

-  **Selección por clasificación**: Después de hacer una selección en la gráfica, podemos usar esta herramienta para que partiendo de los puntos seleccionados, se restrinja solamente a las clases seleccionadas. Este botón es equivalente al de la vista que hemos descrito previamente.

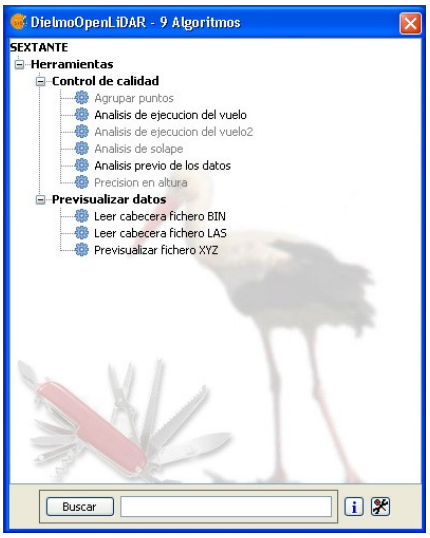
- 

Establecer clasificación: Después de hacer una selección en la gráfica, podemos usar esta herramienta para asignar la clasificación deseada a los puntos seleccionados. Esta herramienta solamente está activa cuando la capa LiDAR está en edición y cuando el nivel de zoom es lo suficientemente alto como para pintar todos los puntos en la vista). Este botón es equivalente al de la vista que hemos descrito previamente.

-  **Imprimir:** Imprime la gráfica longitudinal en la impresora seleccionada.

-  **Guardar**: Guarda la gráfica longitudinal en una imagen en formato png.

Algoritmos LiDAR: Aquí tenemos un recopilatorio donde vamos añadiendo todos los algoritmos de análisis necesarios de los datos LiDAR. En la figura siguiente vemos los disponibles hasta el momento y a continuación hacemos una breve descripción de cada uno de ellos.



- **Previsualizar datos**. Se trata de tres algoritmos que permiten obtener una previsualización de la cabecera de cualquier fichero en formato LAS, BIN o XYZ. Por ejemplo, en la figura siguiente vemos la cabecera de un fichero LAS, donde podemos observar entre otras muchas cosas, el número de puntos, los valores máximos y mínimos de X, Y, Z, etc.

- **Control de calidad**. Estos algoritmos se aplican a los datos LiDAR, obteniendo como resultado ficheros vectoriales con la información necesaria para que un técnico pueda determinar si la ejecución de los vuelos se ha ejecutado en función de las especificaciones:

1. **Análisis previo de los datos**. Este algoritmo coge como entrada un listado de capas LiDAR y genera como salida un fichero vectorial de polígonos en formato shp que contiene un rectángulo con la extensión geométrica de cada uno de los ficheros LiDAR de entrada. Además, para cada polígono se añade una tabla adjunta con la información de la ruta del fichero de entrada, densidad aproximada de puntos por metro cuadrado, número total de puntos del fichero, área del rectángulo generado y alturas máxima y mínima dentro del fichero. Toda esta información se obtiene de la cabecera de los ficheros de entrada, por lo que se genera de forma muy rápida y sirve para tener una primera idea de si están todos los datos.

2. **Agrupar puntos**. Este algoritmo coge como entrada una capa vectorial en formato shp de puntos y una distancia en metros (por defecto 50m). El resultado es una nueva capa vectorial de puntos en formato shp que contiene la misma información que la capa de entrada, al que se le ha añadido un campo nuevo con un identificador de grupo. Todos aquellos puntos cuya distancia entre alguno de ellos sea inferior a la distancia indicada en el campo de entrada se clasificarán con el mismo número de grupo. Los grupos se numeran por orden correlativo desde el nº 1 hasta el número de grupos encontrado en el fichero de entrada. Este algoritmo se utiliza internamente en el análisis de la precisión en altura.

3. **Precisión en altura.** Este algoritmo coge como entrada un listado de capas LiDAR, una capa vectorial en formato shp de puntos con las zonas de control, un desplegable para elegir el campo que tiene la información de la altura en el shp de puntos de entrada y un parámetro numérico con el radio (en metros) con el que vamos a buscar. Para cada punto de la capa de entrada busca entre todas las capas LiDAR los puntos que caen dentro del radio indicado como entrada (suponemos que los puntos de control se han tomado en zonas planas) y realiza un cálculo estadístico con el error medio y desviación estándar, comparando la altura del punto de control con los puntos LiDAR encontrados dentro del radio indicado. Posteriormente, hacemos una segunda selección de puntos, eliminando los extremos que están fuera de una desviación estándar de la media (para eliminar puntos altos o bajos que nos añadan un error a la medida). Internamente se llama al algoritmo de agrupar puntos, para agrupar aquellos puntos de control que pertenecen a las misma zona de control, de forma que se puedan hacer informes de precisión en altura por cada punto, por grupo o por el total de los puntos, por lo que también se añade un campo con la identificación de grupo.

4. **Análisis de la ejecución del vuelo**. Este es el algoritmo con más tiempo de cálculo debido a que se recorren el 100% de los puntos LiDAR apoyándonos en imágenes de la resolución de trabajo indicada. Cuanto menor sea la resolución de trabajo, se obtendrá una mayor definición en la forma de los polígonos obtenidos como ficheros de salida, pero por el contrario llevará más tiempo de cálculo y si la extensión de las capas LiDAR a la resolución de trabajo no cabe en memoria se producirá un error. Lo ideal para el formato que tienen los datos LiDAR del PNOA es trabajar a una resolución de 20x20m. Como resultados obtenemos 4 ficheros vectoriales:

   - Zona volada: Mapa vectorial de polígonos con la zona volada, almacenando un polígono por cada línea de vuelo presente en cada uno de los ficheros LAS que componen el vuelo LiDAR. Además para cada uno de los polígonos se almacena el código de línea de vuelo y el número de puntos LiDAR que contiene.
   - Agujeros: Mapa vectorial de polígonos con las zonas que han quedado sin volar. Estas zonas sin

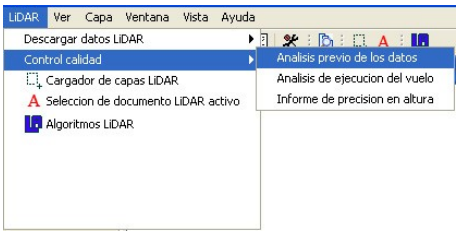
dato suelen ser zonas de agua, pero este análisis nos puede ayudar a detectar si hay alguna zona que se ha quedado sin volar debido a un error del sensor. En este algoritmo se obtiene una capa provisional que sirve como entrada al algoritmo siguiente para obtener el fichero de agujeros definitivo.

- Densidades: Mapa vectorial de polígonos con la densidad del número de puntos por metro cuadrado con un paso de malla de 100m en formato vectorial. Este mapa nos permitirá analizar si se han cumplido los requerimientos de densidad de puntos por metro cuadrado.
- Puntos planos: A partir de los datos LiDAR originales, buscamos zonas muy planas, donde la variación máxima en altura (diferencia entre la altura máxima y mínima) sea inferior a 30cm dentro de la resolución de trabajo. A partir de esta selección de puntos, nos quedaremos con aquellos que caen en zonas de solape y almacenamos un fichero vectorial de puntos con las diferencias de alturas entre pasadas que nos servirá para realizar un análisis del ajuste en altura entre pasadas.

5. **Análisis de la ejecución del vuelo 2**. Este es el algoritmo es continuación del anterior, y coge como entrada las salidas descritas en el punto anterior. Hemos decidido separar este proceso en dos fases debido a que la primera fase tiene un elevado coste computacional a tener que recorrer el 100% de los datos, y si se produce un error en esta segunda fase no es necesario tener que repetir todo el proceso anterior. Como resultado se obtienen otros 4 ficheros vectoriales que describimos a continuación:
   - Ámbito total de la zona volada: Partiendo del fichero vectorial de la zona volada se disuelve para obtener un único polígono con el área de la zona volada.
   - Líneas de vuelo: Partiendo del fichero vectorial de la zona volada se disuelve por el campo que almacena el código de línea de vuelo para obtener un único polígono por cada línea de vuelo, donde se almacena su código, el número de puntos contenidos en dicha línea de vuelo, el área del polígono y la densidad media de puntos en la línea de vuelo.
   - Zonas de solape: Partiendo del fichero de líneas de vuelo, obtenemos un nuevo fichero vectorial con los solapes entre pasadas y generamos una nueva capa vectorial de polígonos con los recubrimientos transversales entre pasadas. En la tabla adjunta se almacena el código de las dos líneas de vuelo que dan como resultado este solape, el porcentaje mínimo, máximo y medio de solape entre dichas pasadas.
   - Agujeros: Termina de generar este producto a partir del ámbito total de la zona volada.

Para que llamar a todos estos algoritmos sea lo más rápido y sencillo, dentro del menú LiDAR que aparece en la vista (lo vemos en la figura siguiente), hemos definido una serie de pasos automáticos para el cálculo del control de

calidad de datos LiDAR.



Al pulsar cada una de estas opciones, se cogen los datos de entrada y salida desde el documento LiDAR y se llama a los algoritmos descritos en el punto anterior de forma automática, de forma que para obtener todos los ficheros vectoriales descritos anteriormente, solamente es necesario definir los datos LiDAR de entrada y la ruta de salida en el documento LiDAR y posteriormente llamar desde el menú LiDAR de la vista a estos tres pasos:

1. Análisis previo de los datos.
2. Informe de precisión en altura.
3. Análisis de ejecución del vuelo.

6.- Más información, actualizaciones, actualizaciones y datos de muestra

En www.dielmo.com puede encontrar:

- Más información sobre la tecnología LiDAR.
- Actualizaciones semanales con nuevas funcionalidades.
- Datos de muestra para poder comenzar a manejar el programa.
- Tutoriales para realizar controles de calidad a los datos LiDAR.
- Previsión de trabajos futuros
- Cómo colaborar con el proyecto, etc.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but

does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License. Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it. However, nothing else grants you permission to modify or
distribute the Program or its derivative works. These actions are
prohibited by law if you do not accept this License. Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions. You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all. For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices. Many people have made
generous contributions to the wide range of software distributed

through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY

YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
> This is free software, and you are welcome to redistribute it
> under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

A. ACRÓNIMOS

B. LICENCIA

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software,

and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors'reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program"means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as ƶou".
Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate

copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your

rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.
In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

> a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to

copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipientséxercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any

particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and .ªny later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free

software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM .ªS IS"WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively

convey the exclusion of warranty; and each file should have at least the çopyright"line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c'for details.

The hypothetical commands 'show wánd 'show c'should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show wánd 'show c'; they could even be mouse-clicks or menu items–whatever suits your program.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> 'Gnomovision'(which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

BIBLIOGRAFÍA

[1] distribución de binarios y código de gvsig. Consellería de infraestructura y transporte (CIT).

[2] Eclipse, descargas y manuales de eclipse. http://www.eclipse.org/.

[3] Sextante, sistema extremeño de análisis territorial. http://forge.osor.eu/plugins/wiki/index.php?id=13type=g.

[4] Terrascan, distribución del software, manuales, tutoriales, etc, de terrascan. http://www.terrasolid.fi/.

[5] Simon Bennet. *Análisis y diseño orientado a objetos de sistemas usando UML.* McGraw-Hill/Interamericana de España, 2003.

[6] Dielmo 3D S.L. [online]. URL: `http://www.dielmo.com/`.

[7] Oscar García González. Desarrollo de software libre para el manejo, visualización y análisis de datos lidar. Universidad de Valencia, 2009.

[8] Portal gvSIG [online]. URL: `http://www.gvsig.org/web/`.

[9] El Mahdi Haloui. Adaptación del formato DWG 2004 a gvSIG. Universidad Politécnica de Valencia, 2008.

[10] The Java tutorials [online]. URL: `http://download.oracle.com/javase/tutorial/`.

[11] JTS Topology Suite [online]. URL: `http://www.vividsolutions.com/jts/JTSHome.htm`.

[12] Latex [online]. URL: `http://en.wikibooks.org/wiki/LaTeX/`.

[13] LIDAR, información sobre esta tecnología [online]. URL: `http://es.wikipedia.org/wiki/Lidar`.

[14] Henry Chang Lo Huang. Aplicación para el aprendizaje de la Geografía con GVSIG. Universidad Politécnica de Valencia, 2007.

[15] Open Geospatial Consortium [online]. URL: http://www.opengeospatial.org/.

[16] Francisco José Peñarrubia. Manual para desarrolladores gvsig v1.1. http://www.gvsig.org/, 2010.

[17] Aurelio V. García Rochera. Proyecto lidar en oliva. Universidad Politécnica de Valencia, 2005.

[18] John Wiley Sons. *Advances in remote sensing and GIS analysis.* Chichester, 1999.

[19] Desarrollo Orientado a Objetos con UML [online]. URL: http://www.clikear.com/manuales/uml/index.aspx.

[20] Eustaquio Vercher Gómez. Desarrollo de un driver de lectura y un writer de escritura para gvSIG de los formatos de MapInfo y MicroStation. Universidad Politécnica de Valencia, 2008.

[21] Jose Vidal Salvador. Creación de extensión para soporte a SLD en gvSIG. Universidad Politécnica de Valencia, 2008.

[22] World Wide Web Consortium W3C [online]. URL: http://www.w3.org/.

[23] gvSIG, información acerca del proyecto. [online]. URL: http://es.wikipedia.org/wiki/GvSIG.

[24] SEXTANTE (SIG), información acerca del proyecto. [online]. URL: http://es.wikipedia.org/wiki/SEXTANTE_(SIG).

[25] Jesus Zarzoso Muñoz. Edición de escenas 3D sobre OpenSceneGraph. Integración en el Sistema de Información Geográfica gvSIG. Universidad Politécnica de Valencia, 2008.

C. AGRADECIMIENTOS

A mis padres por haberme dado todo. Sin ellos no sera hoy quien soy. A Laura que me ha soportado y me ha animado en los momentos que flaqueaban las fuerzas para seguir adelante. A mi hermano y mi cuñada. A Dielmo 3D S.L. y Jose Carlos García por permitirme desarrollar este proyecto que me ha enriquecido tanto personalmente como laboralmente. A los compañeros de Dielmo Oscarín, JV, Aure, Cristina, Carlitos, Rubén, Rosa ha sido un placer trabajar con vosotros.