Dynamic C2 application of Imagery and GIS information using backend repositories

John Hildebrandt, Mark Grigg, Scott Bytheway, Robert Hollemby and Simeon James

Information Technology Division
Defence Science and Technology Organisation
DSTO C3 Research Centre
Fernhill Park
Department of Defence
CANBERRA ACT 2600
AUSTRALIA

POC: john.hildebrandt@dsto.defence.gov.au

Abstract

This paper examines the application of backend imagery and geospatial repositories to provide value added services based on the combination of information from these repositories. To support applications involving short lead times that require fused information products the focus here is on functionality that can dynamically provide the products to the client with minimal manual intervention. Such capability will expand the application of advanced value add products. The imagery and GIS repositories are described followed by an outline of some initial prototype and planned applications.

Keywords

Imagery, GIS, Repository, Visualisation, Fusion, Mosaicing

1 Introduction

Current trends in the use of C2 systems have required them to be rapidly adaptable to new situations and locations. Imagery and Geospatial information are recognised as critical foundation data in any C2 system and organisations such as NIMA are collecting large amounts of foundation data to meet the wide range of situations that are envisaged. However once this raw information is collected various value adding processes are often required to generate specific products to support C2 systems. For example rather than images of point areas or separate imagery and terrain data decision makers may require mosaics of specific broad areas or 3D visualisations as the backdrop for situation and decision sub systems.

Currently several research and commercial systems exist for combining Imagery and Geospatial information for use in applications such as terrain visualisation and situation displays. A difficult problem for many of these systems has been population of datasets and getting the data to client applications in the correct form efficiently. This has limited the application of these tools to specialist areas where the necessary data could be collected and processed. Hence in the typical mode of operation when a new situation arises in a new geographic area a request for new custom products would have to be made to a specialist Organisation that had the skills and tools to combine the raw information. Further these custom products in digital form would be extremely data intensive which may limit their use within the C2 system itself.

Recognising the need to provide a more systematic access mechanism to Imagery and Geospatial data, repositories are being developed to manage these information types. For example the NIMA Libraries program is constructing Imagery and Geospatial Digital libraries that will be accessible via a standard Application Programmer's Interface (API). At DSTO the Image Management and Dissemination (IMAD) system [Grigg *et al*, 1999] [Coddington *et al*, 1998] and the Geospatial Services Segment (GSS) are two instances of experimental imagery and GIS repositories systems that are available in our research environment. Each system will have an API interface to access the functionality and data of the system. Separate groups in DSTO have developed these experimental systems and the challenge is to demonstrate how to make best use of such component systems within a C2 environment.

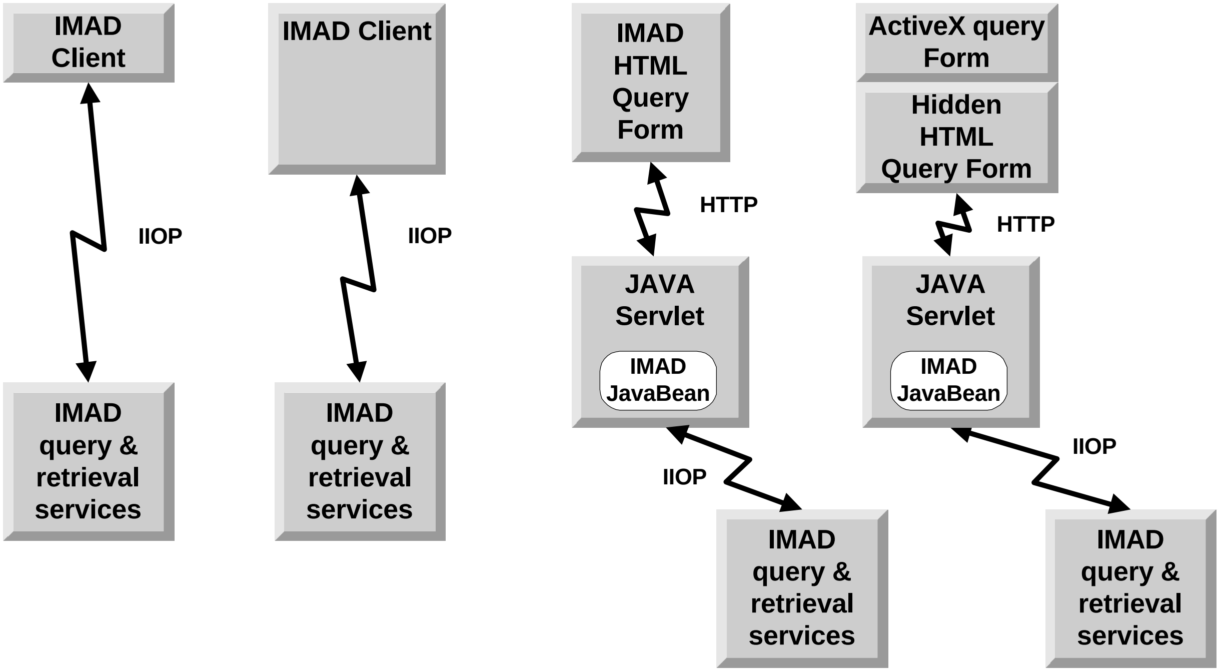
To take maximum advantage of these backend services a collection of client tools and components are required that can access and combine the information sources in different ways to meet the requirements of the end users. This document will examine value-added services that can be built on top of IMAD, GSS, and other services. The focus here is on capabilities that access information directly from the repositories and generate end user views dynamically rather than relying on specialists to fuse the data into specialised datasets (e.g. image mosaics and 3D visualisations). This focus is important since in situations where value added products of new areas are required in short time scales with little prior warning, then the ability to generate these products directly from the source repositories rapidly will be required.

The remainder of this paper is structured as follows, first a brief description of the IMAD and GSS repositories is given, then a number of existing and planned client and server side developments for combining information from each repository are described. These activities have the following benefits; provide value-added services to users of imagery and geospatial information, demonstrate combining components from multiple research areas, and help identify requirements for the further development of the base level components. To support visualisation of large areas and for backdrops to other information displays the dynamic generation of 2D imagery and map mosaics and 3D terrain visualisations is investigated. The fact that these datasets are generated dynamically on demand avoids the generation of extremely large datasets and so makes the use of such visualisations more feasible. This would be achieved by only generating views of the current region being viewed at the minimum resolution required for display. Such functionality makes use of the backend repositories ability to provide imagery and GIS information of specific regions at multiple levels of resolution. Further the use of dynamic servers to generate value added data of areas on demand would reduce the time required to get such data to the C2 system environment. Ultimately it may be possible to serve such visualisations and value added products via a web interface that would greatly simplify training issues. For dynamic display of 3D information in a web environment VRML may be employed.

2 Current services

2.1 IMAD

Currently IMAD provides access to a distributed library of image information via three specialised CORBA services [Siegel, 1998] with interfaces defined in CORBA IDL. The base interface is the library interface that is a subset of the US National Imagery and Mapping Agency (NIMA), Geospatial and Imagery Access Specification (GIAS) specification [GIAS, 1998]. The IMAD system provides extra functionality via two additional services. The query interface or Image Query Manager (IQM) provides access to a distributed collection of libraries and provides a federated query mechanism across the collection of imagery libraries. Each library is assumed to implement the GIAS interface and registers its existence with a CORBA trader, which the query service uses to locate the libraries. Secondly the retrieval interface or Image Dissemination Manager (IDM) uses a factory pattern to generate a retrieval object to get imagery to the client in the most appropriate form. While the CORBA services can be used directly, JavaBeans have been developed to simplify access to these interfaces for developers. The IMAD project has developed an example client application to exercise the repository interfaces. The following figures illustrate the evolution of the IMAD client applications driven both by the IMAD project and the requirement to facilitate integration with geospatial services. Part (a) of the figure shows the IMAD client that interacts directly with the CORBA API's defined as part of the IMAD backend services. In part (b) the use of the IMAD JavaBean is illustrated which simplifies use of the IMAD services and provides a component bean that can be embedded in other applications. In both (a) and (b) CORBA IIOP remote procedure calls are used to the client which to run in a browser will require a digital certificate infrastructure to employ signed Applets. In addition (a) and (b) require communications paths that support IIOP. To provide additional client deployment flexibility and support HTTP only channel options (c) and (d) have been prototyped. In this case the IMAD JavaBean is hosted on the server side in a JAVA Servlet associated with the web server that communicates via IIOP to the IMAD services as before. In these cases only HTTP communication to the client is required and the front end can be provided by a HTML query form or an ActiveX query form. The use of the ActiveX query form facilitated the integration with a commercial ActiveX based mapping component.



(a) (b) (c) (d)

Figure 1: IMAD Client Evolution

2.2 GSS

Currently the GSS is implemented using a vendor solution in order to provide functionality rapidly. It is planned that interfaces based on the work of the OpenGIS consortium will be provided in the future. This will support mixing components from multiple sources.

The Geospatial services are provided by an ORACLE repository with ESRI Spatial Data Engine (SDE) layered on top. This provides a vendor specific API to geospatial data and the intention is to ‘wrap’ this capability with interfaces based on the emerging OpenGIS based standards. To access the repository several options based on commercial products are available. For web based access of predefined GIS data the Arcview Internet Map Server has been setup and allows users to browse Arcview GIS projects using a web browser and JAVA applet. ARCView can be used on a client workstation and connects via the SDE API to the SDE server providing a client server mode of operation. For component development that can access the SDE repository the MapObjects product provides an ActiveX mapping component that can be integrated with client applications.

The first OpenGIS aware API’s planned to be developed are the catalog service to allow the identification of datasets available in the repository. As with the IMAD services a CORBA implementation is intended with JAVABeans being constructed to simplify access to the backend services.

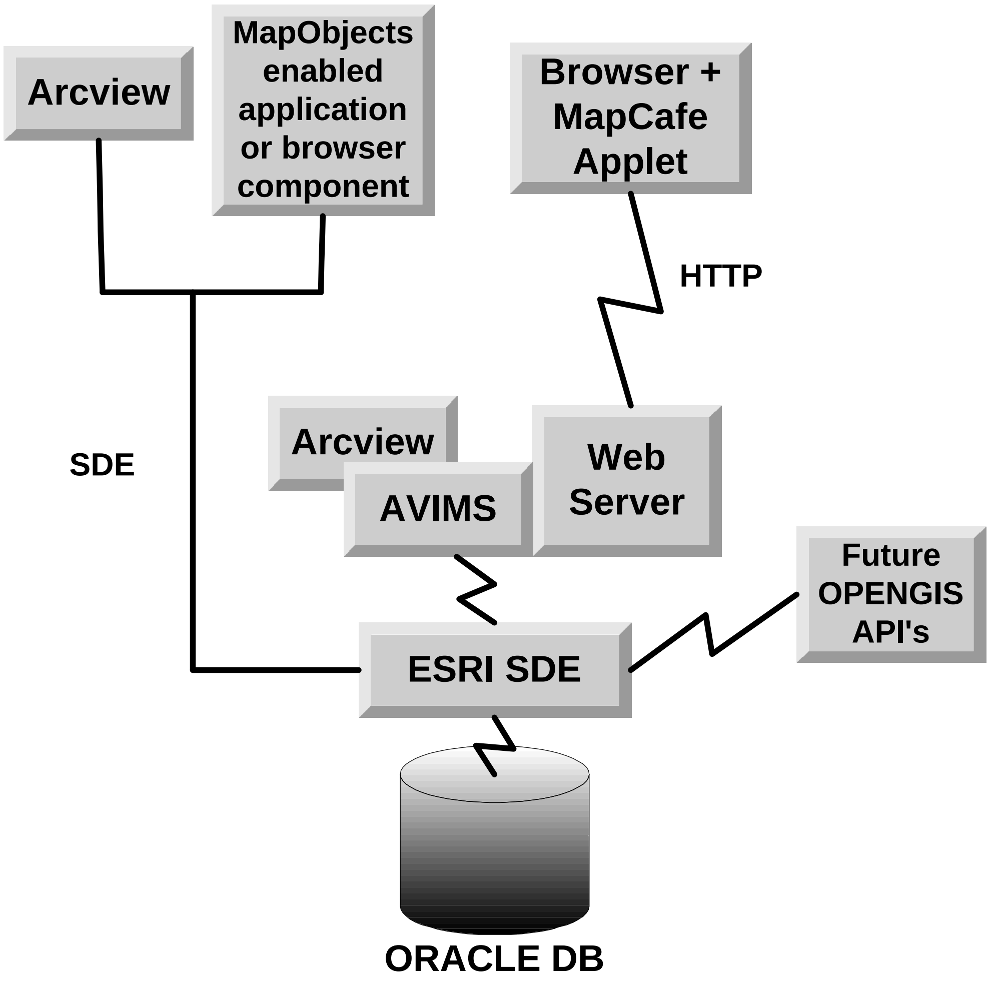


Figure 2: Current Geospatial service

2.3 Client side tools

To develop client side applications that combine both IMAD and GSS functionality, desktop components that reach back to the repositories are required. In the case of geospatial services two options have been investigated. Initially an ActiveX component MapObjects was employed to provide a rapid solution compatible with the initial GSS SDE solution. Now to provide a more cross platform solution the OpenMap JavaBean component from BBN is being used to provide desktop components to access the backend geospatial information. Currently only file-based access to the geospatial information is supplied with this bean but support for OpenGIS API's is planned which fits well with the development plans for the GSS. If required an SDE layer plugin could be built as an interim measure. The OpenMap application is based on a MapBean that loads layers developed as JAVA classes. The MapBean manages the combining of the layers into one-view and sends requests to each of the layers when a redraw is required. For client side access to the IMAD services the IMAD JAVABeans are used as discussed later in this paper.

3 Scenario development

The IMAD project identifies 3 types of user scenarios that guide the development of their system.

1. A Level one user is identified as one who has a low-end machine such as a laptop on which specialised software can be installed for access to the IMAD services. The communication path can be of low bandwidth so that advanced compression mechanisms may be required. The scenario in this case is for the user with this environment to query the IMAD system for imagery and access particular images from the query results.

2. A Level two user is identified as having a standard workstation with LAN access. It is intended that they access the IMAD services via web browser interfaces to avoid the requirement to install software on the clients. So in this case the scenario is to use a browser based query interface and then select images for display using an applet that only sends the minimum amount of data to display the resolution and area of the image being viewed.

3. A Level three user is identified as an Imagery Analyst who uses a high-end exploitation workstation to exploit imagery using digital techniques. This scenario involves the use of a query interface to locate the imagery of interest then an FTP process is invoked to transport the full resolution imagery to the workstation for processing.

In this report further scenarios are identified that employ other backend services other than the IMAD service and support different modes of interaction with the user or application. It is intended that these scenarios will be used to guide development of specialised clients that combine information from the backend services in different ways, and to suggest possible extensions to the backend services. The scenarios are now listed and some expanded upon in latter sections.

4. Scenario four deals with the use of geospatial services to provide a graphical geospatial front end to the IMAD service. So here the scenario is for the user to interact with a graphical GIS front end to perform the same types of queries as described in the three IMAD user levels. This can be divided into two steps:
   (a) A Map based interface is employed to input the regions of interest for an IMAD query;
   (b) A Map based interface is employed to output the results of an IMAD query showing the image hits list on a Map.
   Both (a) and (b) can be combined into a single interface as has been done by the IMAD project.

5. The next scenario proposes treating the image coverage's as a standard dynamic GIS layer rather than a 'special' layer generated by special steps via user interface based queries. For example if a user wishes to view rivers on a GIS display they typically simply turn on the river layer for display rather than submitting a query for rivers within a certain geographical extent. Hence image coverage's could be such a layer that the user could turn on when they wish to view imagery coverage's on the GIS display.

While the three IMAD user types could employ such a scenario this will stress the backend services in a manner different than in scenarios 1,2, and 3. In scenario’s 1,2, and 3 the user inputs a relatively small number of queries, then the hit list is produced for the user to select from. In this scenario however an IMAD query would be needed each time the geospatial display is altered by the user that in scrolling and zooming situations could generate large numbers of queries. As in scenario 4(b) selecting an image coverage could be used to initiate display of that image.

To be geospatially accurate the coverage polygons should reflect the true ground coverage of the imagery. Hence knowledge of the imaging geometry and mapping projections used will be required.

6. In this scenario it is proposed that the user treat the imagery itself as a standard GIS layer. This may be required to support viewing of broad regions rather than just individual images separately. In this case the interaction will be similar to scenario 5 but instead of image coverage’s being displayed the actual image data is displayed. To avoid the client being swamped with data only the required imagery data for a particular region or resolution should be transmitted, as IMAD is designed to support. In this case multiple images will be rendered onto the GIS layer. To handle imagery overlaps some layer management within this GIS layer will eventually be required. To be geospatially accurate the image data will need to be registered to the map projection before being rendered onto the GIS image layer.

7. In this scenario it is recognised that the GIS layers will also include 3D information that the user may wish to visualise in 3 dimensions. For example if Digital elevation data is available the user may wish to view imagery draped over the elevation data. So as in scenario 6 it is desired to treat imagery as a dynamic GIS layer but a different 3D display mechanism will be employed. Much of the same registration software should be reusable from scenario 6 but a 3D-display representation will be generated.

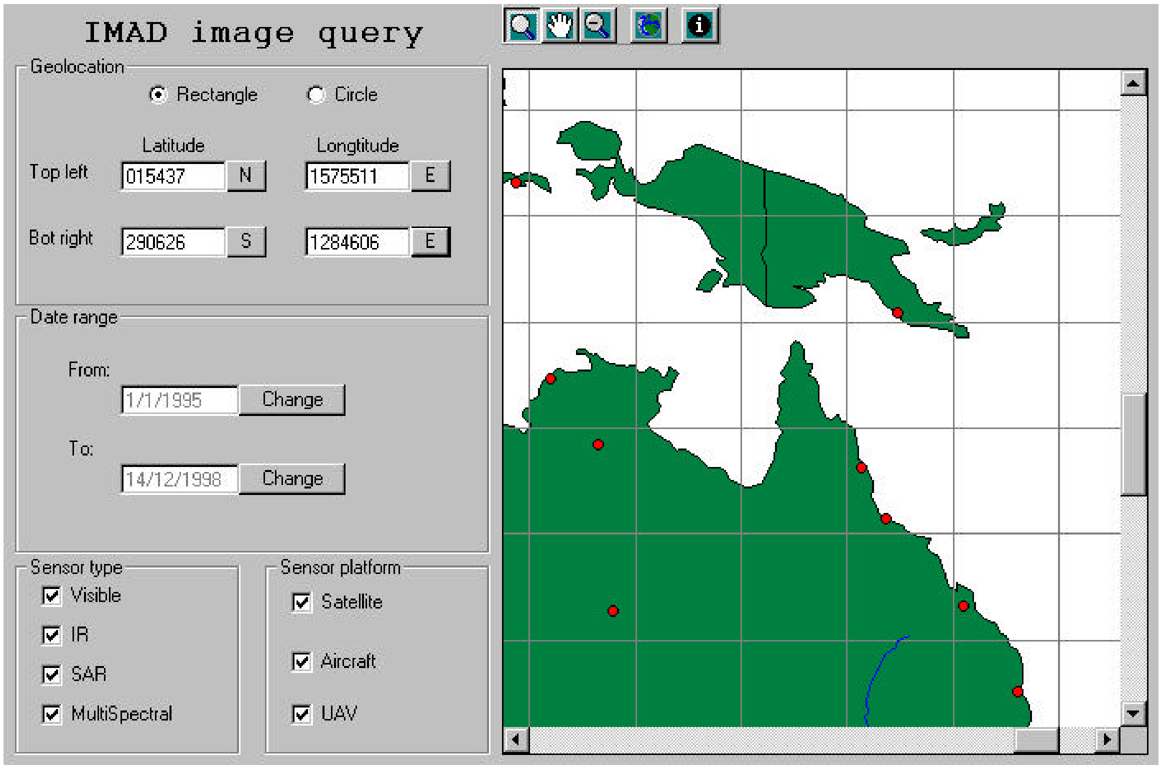
8. In this scenario we propose that two image analyst or users wish to collaboratively interact with the same image. This would involve both users annotating the image while in audio or videoconference with each other. If the users are image analysts they may also wish to apply exploitation algorithms to the image collaboratively. In this scenario the system should minimise the need to transport pixel data between the sites. If each user has access to the same image in local IMAD repositories then the imagery should be sourced locally with only overlay information and processing instructions disseminated between the users.

9. In this scenario we propose an environment where an image analyst is using an exploitation tool that can source imagery directly from an online repository rather than having the data transferred via FTP to the client workstation. In this case the analyst requires accurate pixel data but is using an exploitation framework that supports on demand data delivery and processing. Such a framework introduces the possibility of distributed processing of imagery.

Other possible applications are also described in brief in latter sections.

4 GIS based selection of region of interest for IMAD query client

The IMAD query client previously required a user to enter numeric data to select the region of interest. A small MapObjects ActiveX component has been constructed that displays a map drawn from the GIS services (currently for testing local GIS data is being used; however the mapping component supports connection with the backend repository without modification). This component allows the user to select a region of interest and would then send the numeric values to the IMAD client query component (so implementing scenario 4(a)). The component can be embedded in a web page in conjunction with a HTML form based IMAD query form or used as a helper component with the IMAD query component. The outcome of this allows map based queries for images and demonstrates two separately developed services (IMAD and GIS) being used together. The results of the query will be the standard IMAD 'hit list' either in textual form as is the current case or in graphical form as a overlay of image coverage's on a GIS background. Figure 3 illustrates the interface that provides graphical geospatial query of the image repository. Figure 4 shows the interaction of the client level components, and the backend repositories.


IMAD image query

Geolocation
● Rectangle ○ Circle

	Latitude		Longtitude	
Top left	015437	N	1575511	E
Bot right	290626	S	1284606	E

Date range
From: 1/1/1995
To: 14/12/1998

Sensor type
☑ Visible
☑ IR
☑ SAR
☑ MultiSpectral

Sensor platform
☑ Satellite
☑ Aircraft
☑ UAV

Figure 3: IMAD query component using GSS client component to select geospatial extent

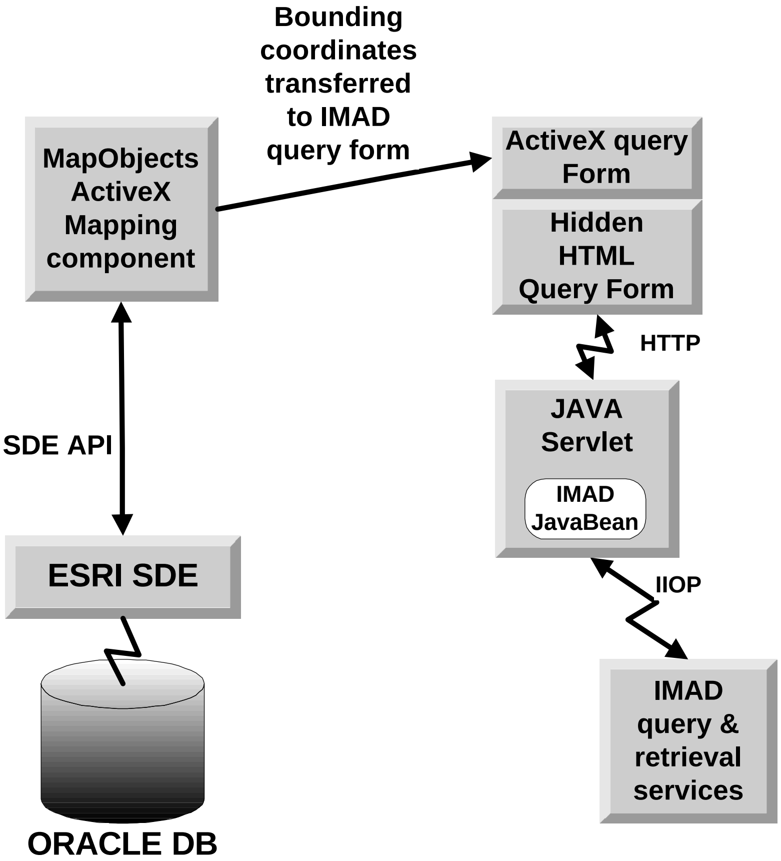

Bounding coordinates transferred to IMAD query form
MapObjects ActiveX Mapping component
ActiveX query Form
Hidden HTML Query Form
HTTP
SDE API
JAVA Servlet
IMAD JavaBean
ESRI SDE
IIOP
IMAD query & retrieval services
ORACLE DB

Figure 4: Use of Geospatial services in Imagery query

Currently only rectangular and circular geospatial queries are supported since the initial goal was to provide a graphical geospatial query front end to the existing IMAD query interface. Once a GIS based front-end is available more complex geospatial queries can be supported. For example selection of polygons and selection of a region near a curved path could be supported by the user interface. Such a change would require change to the IMAD query to support passing more complex geospatial regions to the backend query service. In addition the backend query service does not yet implement query on more complex regions in its API which would need to be corrected to support such a feature. This provides an example of new capability facilitated by combination of services and new user interface paradigms driving the development of the backend service API.

5 IMAD Openmap based client

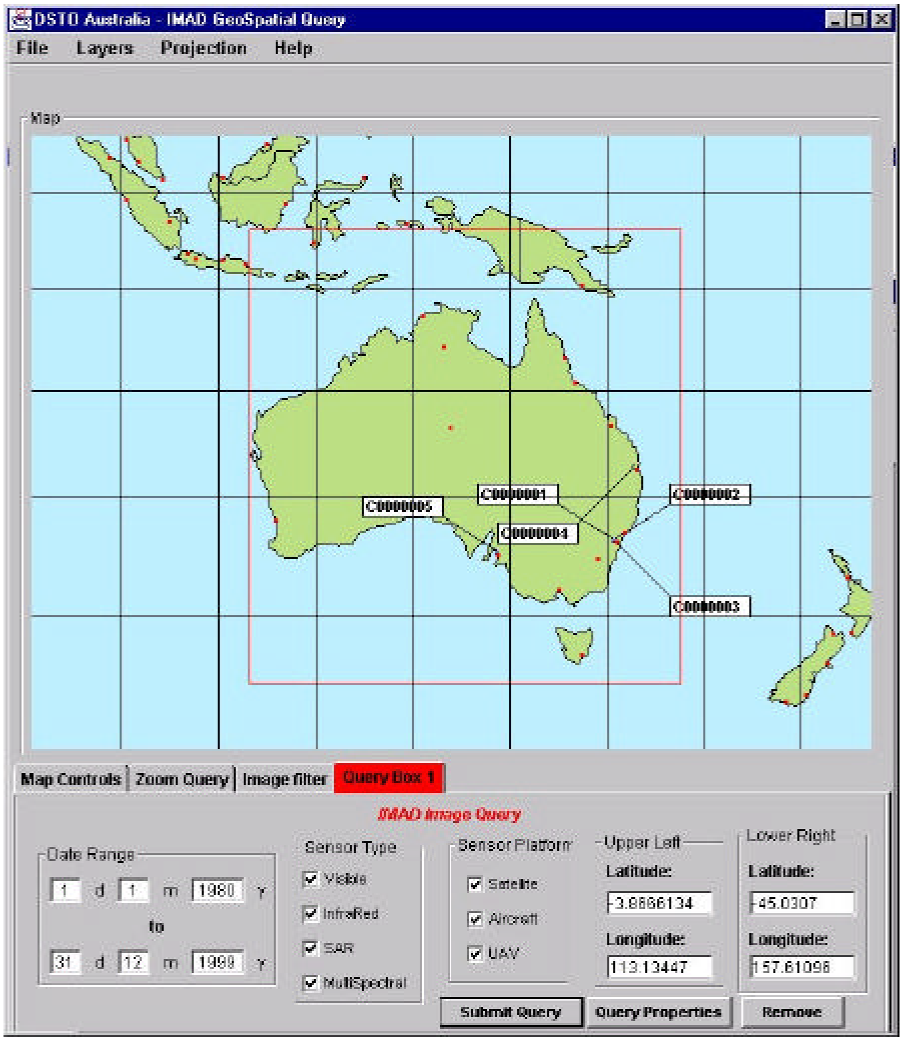
The client application has also been built using Java by combining a geospatial map interface and the IMAD JavaBeans inside a container application, figure 6. The geospatial map interface is based on the Mapbean component available from BBN technologies [BBN, 1999]. The lower part of the client panel contains a user interface that allows modification of the IMAD JavaBean properties. In this case the IMAD JavaBean interface has been hidden and the entry fields and checkboxes are used to modify the properties of the IMAD JavaBean prior to the submission of the query.

The MapBean allows the user to perform actions such as zooming in and out on selected regions and changing the map projection. This component provides for the management and display of a number of geospatially referenced layers. The addition of user defined layers is a relatively simple process and the image query results are in fact displayed in a separate layer. The underlying map data used in the current implementation is loaded from shapefiles in the local file system however this data will eventually be loaded from a backend Geospatial service in a future implementation.

The operation of this client is similar to that in the previous section with the user selecting the region of interest on the map display indicated by an outlined area as shown. Once the area is selected an IMAD image query entry form is created and displayed into which the user can enter additional query parameters for the image. The geolocation information is automatically entered in the query form. This particular client can support up to five simultaneous queries covering different geographic regions.

Once the query has been specified the user then submits the query. This process occurs via the IMAD JavaBean that makes an IIOP connection to the Image Query Manager service. The results of the query are displayed in a layer on the MapBean as a set of flags associated with the images. While it is not clear on this diagram, the area of coverage of the image is also displayed. The selection of an image for display is achieved by ‘clicking’ the mouse over one of the image flags. This makes a connection to the retrieval service through the IMAD JavaBean and creates a display window on the client for the image display.

This client offers the advantage over the previous client in that it is a complete Java implementation and thus not tied to any specific platform. The MapBean component is under continual development and will eventually offer a CORBA interface allowing easy integration of backend geospatial services in much the same way as the imagery services are accessed via the IMAD JavaBeans.



Date Range: 1 d 1 m 1980 y to 31 d 12 m 1999 y

Sensor Type: ☑ Visible ☑ InfraRed ☑ SAR ☑ MultiSpectral

Sensor Platform: ☑ Satellite ☑ Aircraft ☑ UAV

Upper Left — Latitude: -3.8886134 Longitude: 113.13447

Lower Right — Latitude: -45.0307 Longitude: 157.61098

Figure 5: IMAD Map based client

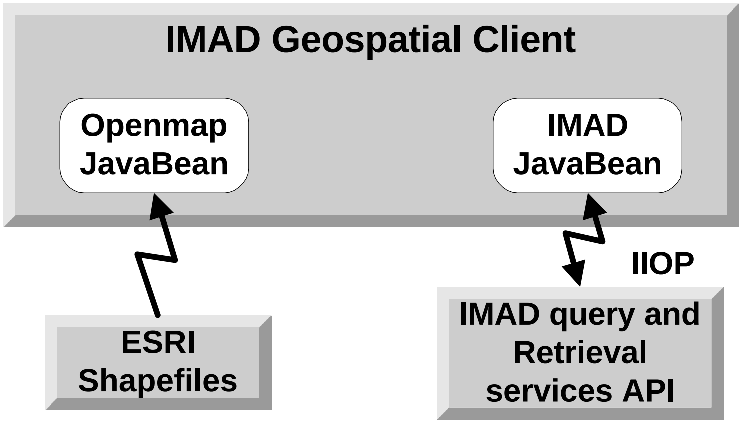


Figure 6: Openmap based geospatial client and IMAD connection

6 Dynamic merging of imagery and GIS data into virtual mosaics

Up to this point an implicit assumption has been that the user would eventually select a single image or set of images for viewing or analysis, that they would use as separate products. However if the focus of the user is to examine a particular geographic region then a better mode of operation may be to provide all the imagery (or image coverage's) to the user on a geographic backdrop for orientation, as described in scenarios 5 and 6. An example use of such technology would be in searching a broad area for objects or activity. Tools are available for mosaicing images together but if each individual image is already a large dataset then mosaicing them only compounds the data handling problem. Rather a preferred approach is to provide an Active server [Hawick, 1998] that can provide the appearance of mosaiced imagery without physically creating such datasets.

The approach here is to consider a GIS based canvas as the background and coordinate system onto which imagery is overlaid on demand as the user pans and zooms spatially. This would provide the advantages of combining multiple images into broad areas without the need to generate large files containing registered and mosaiced images. This service would require a registration service in addition to the IMAD and GIS services.

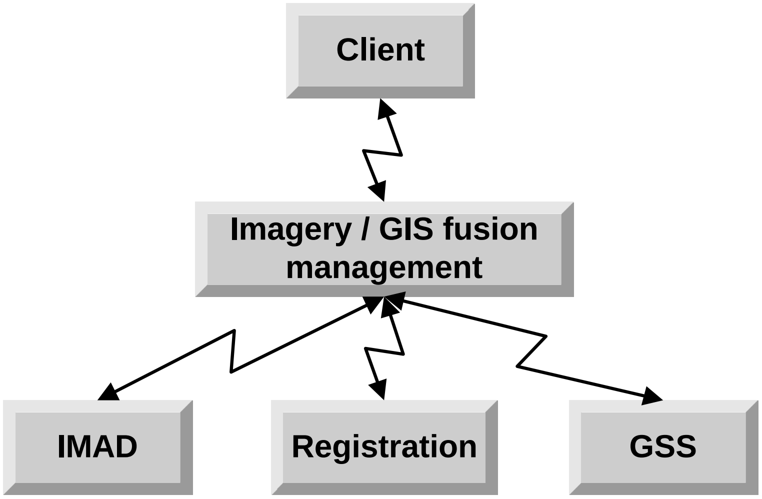


Figure 7: Imagery fused onto geospatial canvas

The previous figure illustrates the basic components that such capability would require. The client component would interact with an imagery / GIS fusion and management component (this could run with the client) that would be initialised by the user selecting the starting region of interest. The management server would then request geospatial data for this region at the appropriate resolution for display and query IMAD for imagery that lies within this geospatial region, again retrieving only the resolution required for display at the time. Before overlaying the imagery on the geospatial display it will need to be registered to the same projection as the geospatial data. To minimise the flow of data it may be advantageous to have the registration service on the same server as IMAD or at least connected by high-speed communications.

As the user pans and zooms the display the interactions described above would need to repeat dynamically to update the display with the correct region and required resolution. Smart cache on both the client and/or server side may be required to maintain acceptable interactive performance. Preemptive and predictive caching would be useful, as the user is likely to view regions surrounding their current viewing location and panning direction can provide predictive hints. Since this application would be requesting data at interactive rates in response to user input it has the potential to make substantial demands for information. Such an application will place significant performance demands on a system such as IMAD in which the imagery may be distributed at multiple locations. So in addition to caching some policies of not using imagery beyond some distance (in terms of communications cost) may be required.

Layer management of the displayed imagery will be required since some imagery may overlap each other due to collections at different times or due to different types of imagery being collected. If only one type of imagery was required this could be handled at the initial query for the data but the overlap within the same imagery type will remain.

Initially to demonstrate the concept of this system pre-registered imagery could be used to remove the requirement of the registration server. It should be noted however that this is an important component to make this a manageable system in that it removes the requirement to have a time consuming registration problem at image ingest. A completely general automatic registration service is still in the realms of computer vision research. However for the case where particular classes of imagery are used that have known imaging geometry and only accuracy sufficient for interactive viewing is required, then existing algorithms and tools should be adaptable to the task. In the case that the user requires accurate position data from the imagery a mapping back to the original imagery would be required.

6.1 Implementation

It was decided to employ the Openmap client to prototype this application to take advantage of the experience gained from the IMAD map client and to access other planned efforts. The project group providing the geospatial repository is planning to develop an Openmap layer to display map data from their repository. It is intended that this layer will try to make intelligent heuristic decisions as to what level of detail of map data to display for a given zoom state. So in our prototyping we are assuming that the map background will be handled by this layer and need not discuss this further here. A file-based map will be used as an interim measure pending this layers development.

With this environment our task in developing the capabilities described in scenario’s 5 and 6 is now reduced to developing special purpose Openmap layers. The layers must display imagery coverage’s (scenario 5) and actual image data (scenario 6) with the geographic extent of the query being the currently viewed area, rather than a separately chosen region as was the case in the IMAD Map client. The implementation arrangement is illustrated in the figure 8.

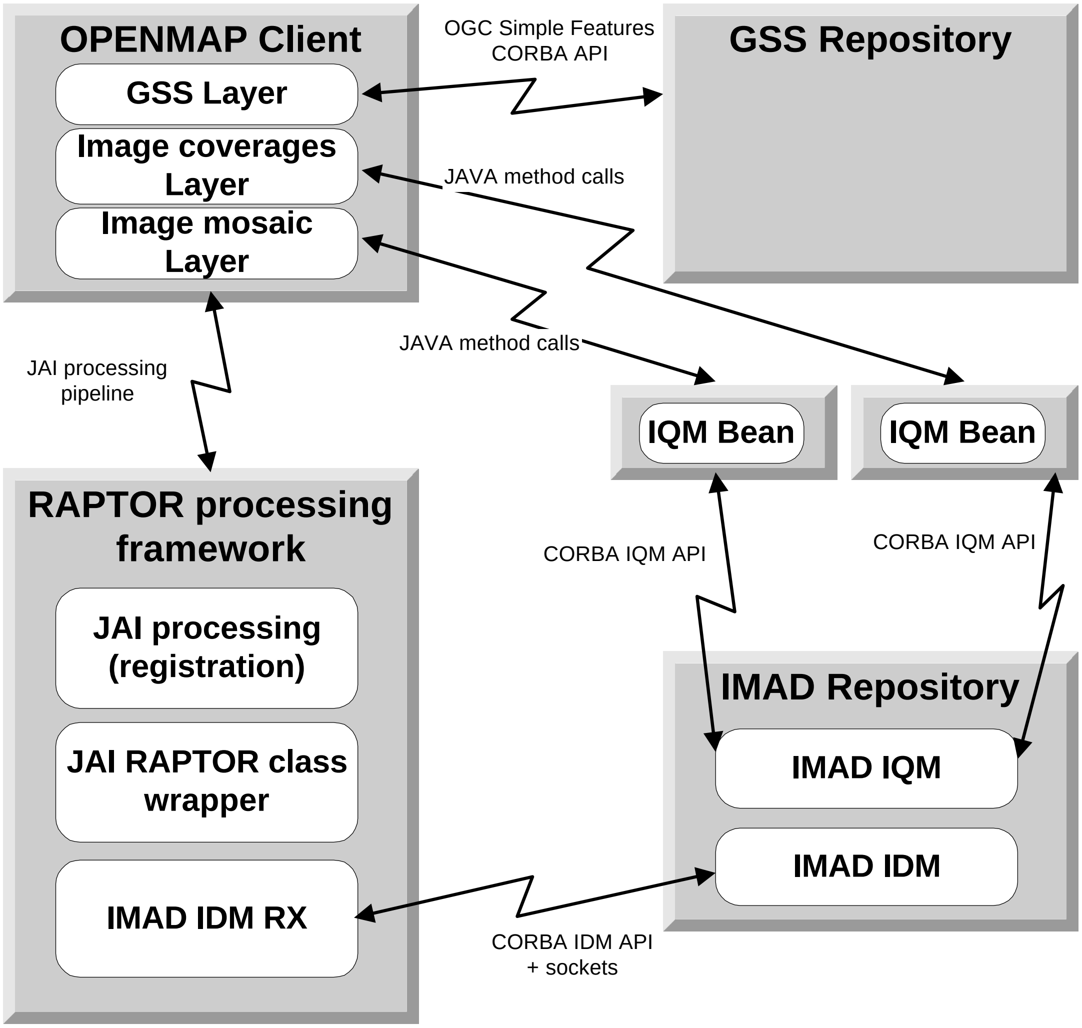


Figure 8: Dynamic mosaic and coverage display capability

The processing of the imagery data would require an environment that could pipe image data from the IMAD repository through various processing algorithms before being rendered on the Openmap layer. Sun are developing the Java Advanced Imaging (JAI) library [JAI, 1999] as a standard JAVA extension to support JAVA based image exploitation application development. The JAI supports several concepts useful for image exploitation frameworks such as tiled images, just in time processing, client-server execution, and processing pipelines. Several formats are supported by the JAI including TIFF, JPEG, Flashpix and the Internet Imaging Protocol (IIP). New formats can be added to the framework and then used as native JAI data types.

Within the JAI framework the PlanarImage class provides the typical base level of access to image data. This class provides access methods for obtaining image data and metadata without concern of the original type of imagery. The JAI framework supports distributed processing of imagery leveraging the Remote Method Invocation (RMI) of JAVA. JAI defines a RemoteImage class to support this capability and once registered with RMI remote users can access this object to perform distributed imaging operations.

To develop the image layer capability we required a mechanism to access the IMAD repository and then apply registration algorithms to retrieve imagery to position it at the correct location on the geospatial layer. This capability is being provided by a general capability being developed to provide closer connection between the repositories and exploitation and visualisation systems that require rapid access to the contents of these repositories. The system called RAPTOR (Rapid Analysis, Processing and Transformation from Online Repositories) in its initial form consists of a JAI class wrapper around the IMAD IDM Bean that provides the transport of the image data from the IMAD repository. With this design the existing IMAD dissemination management objects are used to deliver data to the JAI wrapper that provides a JAI tiled image source. The object wrapper handles the conversion of IMAD image data to JAI tiles to support JAI processing pipelines. Once available as a JAI image object arbitrary JAI processing operations can be applied to the imagery and then displayed using standard JAI display code.

To add such a new image type within the JAI framework, that provides tiles on demand from the IMAD source required the development of a subclass (named RAPTORImage) of the JAI PlanarImage type. The RAPTORImage constructor sets up the connection with the IMAD IDM service using the IMAD ImageClientBean and initialises the JAI image parameters. Inputs to the constructor are the ImageID, Library Address, image width, and image height that would typically be sourced from a query on the IMAD repository.

The other key method that must be implemented is the getTile() method that returns requested tiles of the image. In the case of RAPTORImage this is implemented with the aid of the IMAD Receiver object that handles the actual data transmission. Caching of tiles is provided within the JAI framework that helps minimise retransmission of previously displayed tiles. Further caching is implemented within this subclass to avoid retransmitting data from the IMAD repository in cases where the JAI cache is flushed. The tile size of the RAPTORImage object is defined within the object constructor and can be selected at run time. The underlying IMAD system also employs tiling but the JAI tile size is not restricted to equal the IMAD tile size implementation. An issue for future research is the efficiency aspects of determining the optimal tile size mixes.

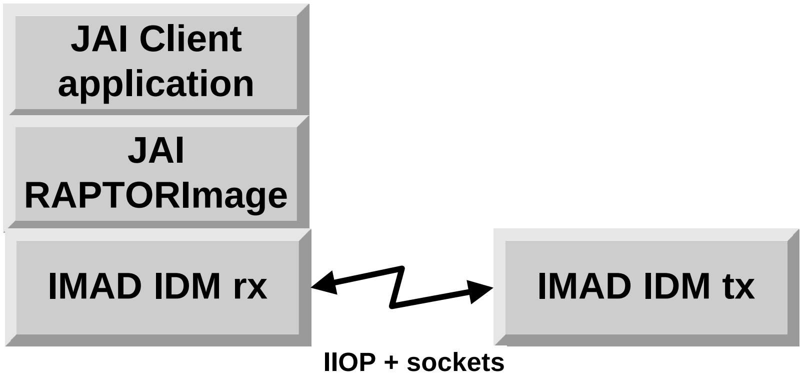


Figure 9: Client side JAI wrapper

With the aid of the RAPTORImage class imagery can be accessed and processed within the JAI framework without any requirement for code specific to the underlying repository. Hence the same client code can be used with imagery sourced from the local file system, a remote JAI image, an IIP server, or the on-line IMAD repository. Since the JAI extensions are available the client applications can apply JAI image processing operators to the imagery without needing to adjust the code for the imagery source. An example client (RAPTORClient) for basic image display has been developed to exercise the RAPTORImage class.

Existing non-JAVA image-processing code can be integrated into this framework via the use of the JAVA Native Interface (JNI). However if the code assumes access to the full image it could reduce the efficiency of the image pipeline concept.

Providing the client side wrapper allows processing of the imagery using JAI on the client system with IMAD protocols used to transfer data as required. An alternative method of providing JAI access is to provide an implementation of the JAI client server imaging capabilities. In this case the JAI remote image class constructor on the client accepts a server reference and a reference to an image or processing sequence that can be instantiated on the server. So in this case the underlying IMAD repository is not required to manage the image dissemination. The JAI framework using the JAVA RMI protocol manages the connection between the JAI client remote image object and the remote Image implementation on the server. Hence the repository (IMAD or other) is merely required to pass an image reference to the processing server which then manages the on demand processing and transmission of the image data to the client. This file reference could simply be a reference to the internal IMAD file store or the IMAD system may need to transform the internal format to one compatible with the JAI framework. The image reference could also be a reference to an IIP server, which is supported by the JAI framework, and supports multi resolution tiled data transmission. The JAI remote image could be instantiated on the same machine as the IMAD library to conserve network bandwidth or on another server that could be optimised for image processing operations. This server side implementation of the JAI framework will now allow image-processing operations to be distributed over multiple platforms. Processing operations will be able to be performed on the platforms best suited to their execution. It is planned to investigate this option in future work.

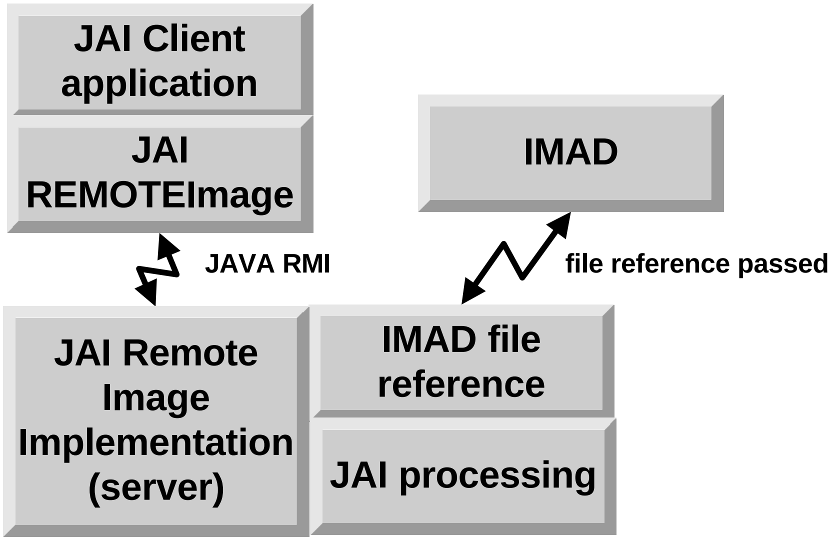
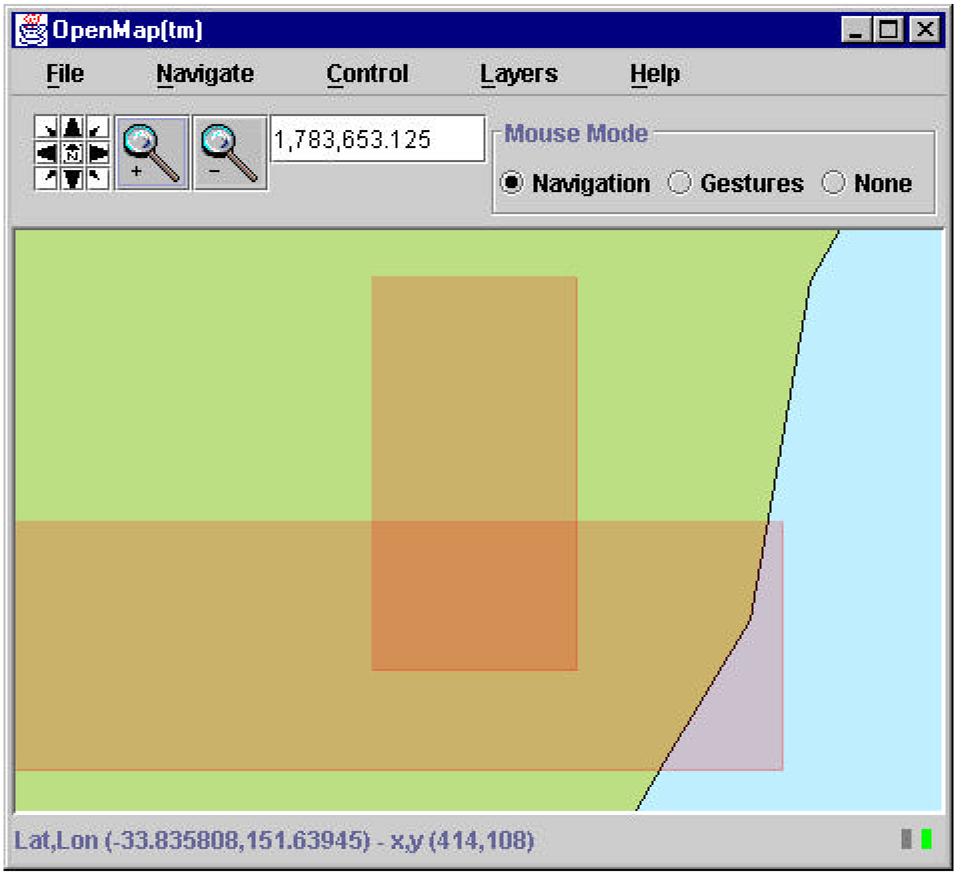


Figure 10: Server side RAPTOR configuration

With the imagery now accessible to the JAI framework the transformation operations available within the JAI library can be used in conjunction with the image metadata to project the imagery onto the Openmap layer. The image coverages layer which retrieves the required metadata and display coverage's has been prototyped, as has the JAI client side bridge. The final step of developing the imagery layer that combines these two developments in currently under development. The image coverage layer is illustrated in figure 11.



1,783,653.125

Lat,Lon (-33.835808,151.63945) - x,y (414,108)

Figure 11: Image coverage layer display

7 Dynamic merging of imagery and 3-D GIS data into virtual 3-D mosaics

Currently many products allow imagery and geospatial information to be combined into 3D datasets. However this has to be done on a case by case basis with a specialist user extracting the required information from the raw information repositories. If such value-added datasets could be generated on the fly then they could be made available to a broader user community and used as a background for other information displays. This is an extension of the last service but here a 3-D display is delivered (scenario 7), possibly using VRML, Java3D or some other 3-D display environment. A 2-D display may be required in addition to help the user maintain orientation due to the more complex nature of navigating 3D information spaces. Again in this case the level of detail transferred to the client must be controlled to avoid transferring excessive data to the client. One method to provide the feedback from the client would be to use the scripting and messaging capabilities with VRML 2.0, if this was used as a display mechanism. The use of VRML would also provide assistance in the level of detail issue by exploiting the Level of Detail LOD nodes in VRML [Carey, 1997]. The GeoVRML working group [GeoVRML, 1998] provides a number of resources which would contribute to this application. For example tools to generate and manage multiresolution datasets could be modified to accept data directly from the repositories rather than from individual files.

The next diagram shows the major components in delivering such a capability.

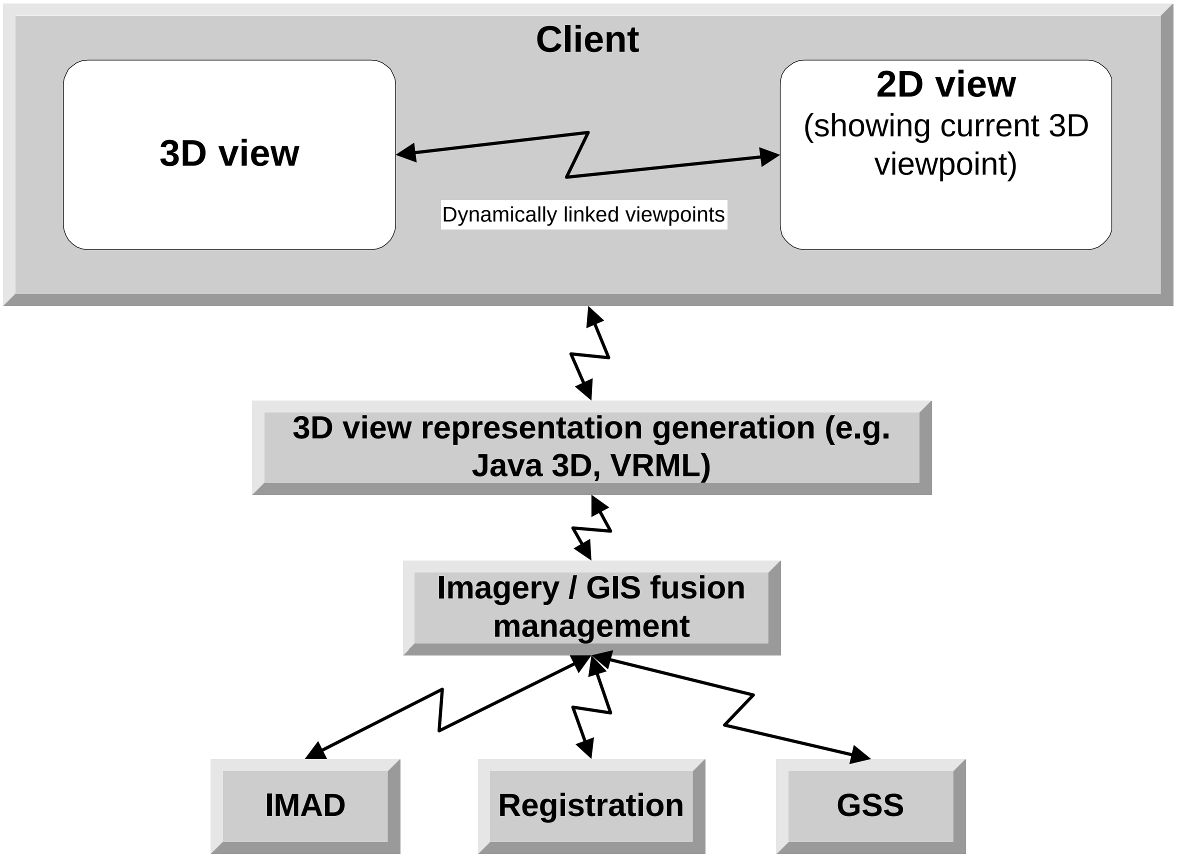

Client
3D view
2D view
(showing current 3D viewpoint)
Dynamically linked viewpoints
3D view representation generation (e.g. Java 3D, VRML)
Imagery / GIS fusion management
IMAD
Registration
GSS

Figure 12: Image GIS fused visualisation with 3D display

In Leung and Coddington (1998) a system has been described that generates 3D visualisations from a single terrain and image collection. However in our case we desire a more scalable solution where the source imagery and terrain data is retrieved as required from the source data repositories. This provides an Active Repository [Hawick, 1998] that is capable of supplying a new data form, namely 3D visualisations on a dynamic as required basis. Such a scheme avoids the requirement to pre-construct value added datasets and so will support situations where value added datasets are required of new areas with little prior warning.

8 Updating GIS data from Imagery

Remote Sensed Imagery is in regular use as a source of information for generation and revision of geospatial information. In addition to mapping production systems for producing large quantities of geospatial data, there is a requirement for tools that allow users that analyse imagery to flag changes that relate to the geospatial data. For example, if a road marked in the GIS can be seen on imagery as having been relocated or blocked then this should be reflected in the geospatial information repository.

Geospatial information is typically updated by authorised mapping experts, so in this situation the updates to the GIS would have to be implemented via annotations to the geospatial information rather than changing the geospatial information itself. So for example an annotation layer or an annotation metadata tag for features could be used to store the updated information. Tools for input of annotation information or displaying the annotation will need to connect to the underlying annotation repository. At least three types of processes would need to be supported. Firstly one needs tools to allow analysts to annotate changes in geospatial information while viewing imagery and geospatial information. Secondly users viewing geospatial information will need the annotated changes made visible to them, and may require links to the source imagery which was the supporting evidence. Finally mechanisms to notify the geospatial data producers of change annotations will be required for input into the production process. Once updated geospatial product is available the management of the annotations against multiple versions of the geospatial data will need to be addressed.

9 GIS overlays and capture of analyst annotation

The previous applications focused on placing imagery onto a geospatial foundation. For users that are focused primarily on Imagery based information, overlay of geospatial information overlay will still be of value, but due to the different focus users may want geospatial information overlaid on the original image data rather than registering the imagery onto a geospatial backdrop. For example concerns as to the accuracy of the image data and preserving this accuracy may drive one in this direction.

Such an application will require the capability to adjust the projection of the geospatial information to match that of the imagery rather than registering the imagery to the geospatial foundation. This capability could be employed to provide a user with supporting geospatial information to facilitate understanding of the imagery.

When analysing imagery, a user will often annotate the imagery to indicate the result of their analysis. Currently such annotation information is embedded in the image product so that the value-added information is not captured in an accessible way within any data repository. Providing an overlay linked to a geospatial repository could enable the ability to capture analyst annotation into a geospatial information repository for latter use. For example if a user identifies an object in the image the location can be captured simultaneously with the identification and stored in the repository. Such a scheme would support latter queries for imagery on the basis of the presence of objects within the imagery without requiring object recognition schemes to be developed. The framework developed would support latter development of object recognition algorithms that would assist in the annotation process.

10 Collection prediction

Collection and requirements managers require tools to assess what imaging collections are possible using different platforms and during different times. Users require tools to assist them in deciding what imagery is required for some complex tasks. For example processing of images into a value added product might require a collection of imagery with particular imaging characteristics. Commercial tools such as satellite orbital display tools can be used to display the paths of the collectors and used to predict future collection possibilities. If the collection constraints involve geospatial information then linkages between the collection modeling and geospatial repository would be required. For example terrain features have a significant effect on collection geometry requirements and so must be factored into any collection modeling activity.

11 Other INT integration and planning applications

Other geospatially-indexed intelligence can be displayed once it is available. In addition with appropriate links to planning information the foundation imagery and GIS data could be used as a backdrop for planning applications. The exact nature of such tools and the required functionality would be highly dependent on the particular information sources.

12 Design Pattern and granularity Lessons

From the previous discussions one can see two classes of components and services emerging. On the client side a variety of small components that can tightly interact with each other and access backend services as required. These components will often have some user interface and so must fit in with an existing interface structure. Typically methods to implement such fine-grained components include ActiveX, web technologies and JAVABeans. In our case it also included layers which can be considered as components within a client application framework (Openmap). On the backend the nature of the components is different with both fine grained and large grained components possible. In addition a much wider range of server types than client systems will exist due to the requirement to connect to a range of data repositories and specialist processing systems. Hence in this part of the environment components and services that communicate via middleware that support heterogeneous environments is required. Example methods include CORBA, and Java RMI. To connect these two component domains a variety of mechanisms can be used with web technologies a popular choice.

13 Discussion

In this paper a number of tools that exploit component based development and the existence of data repository services are described. An initial integration of Imagery and Geospatial repositories is described where a graphical geospatial query mechanism is provided to the Image repository system. The development of a dynamic multi-image display capability was described, and the initial implementation outlined. A number of possible ways to further exploit the integration of the two repositories are described with potential designs of solutions provided.

For the implementation and planning of such services it is seen that the integration of several technologies appropriate for different parts of the overall system is required. For example to provide distributed services in a heterogeneous environment both CORBA and JAVABeans are employed. At the client side the need to fit in with deployed technology and available commercial components leads to the use of finer grained components including technologies such as ActiveX, JAVABeans, and Web technologies.

A number of extensions to the IMAD system would facilitate the further development of the described applications. The query system needs to support non-rectangular queries to support polygon region queries. A client or exploitation framework needs to be able to retrieve more detailed metadata to support geo-processing of the imagery.

14 Bibliography

[BBN, 1999] BBN OpenMap JavaBean, http://openmap.bbn.com.

[Carey and Bell, 1997] Carey, R. and Bell G. *The Annotated VRML 2.0 Reference Manual*, Addison Wesley Developers Press 1997.

[Coddington et al., 1998] Coddington, P.; Hawick, K.; Kerry, K.; Mathew, J.; Silis, A.; Webb, D.; Whitbread, P.; Irving, C.; Grigg, M.; Jana, R.; and Tang, K. *Implementation of a Geospatial Imagery Digital Library using JAVA and CORBA*, DHPC technical Report DHPC-047, Adelaide University, Published in Proc. Of Technologies of Object-Oriented Languages and Systems Asia (TOOLS 27), Beijing, Sept. 1998.

[GeoVRML, 1998]: The GeoVRML Working group, http://www.vrml.org

[GIAS 1998]: "Geospatial and Imagery Access Specification", US National Imagery and Mapping Agency, http://www.nima.mil/aig/products

[Grigg et al., 1999] Grigg, M.; Whitbread, P.; Irving, C.; Lui, A.; and Jana, *Component Based Architecture for a Distributed Library System*, accepted for Distributed Multimedia Systems Conference 99, Aizu, Japan, July 1999.

[Harwick and Coddington, 1998] Hawick, K. and Coddington, P. *Interfacing to Distributed Active Data Archives*, DHPC Technical Report DHPC-043, Adelaide University, to appear in Int. J. on Future Generation Computer Systems.

[JAI, 1999]: Sun JAI web site, http://java.sun.com/products/java-media/jai/index.html.

[Leung and Coddington, 1998] Leung, A and Coddington, P. *Interactive viewing of 3D terrain models using VRML*, DHPC Technical Report DHPC-043, Adelaide University, Publish in Proc. of Asia Pacific Web (APWeb) 98, Beijing, Sept 1998.

Siegel, J. (1998): “OMG Overview: CORBA and the OMA in Enterprise Computing”, Communications of the ACM, Vol. 41, No. 10, pp. 37-43, Oct. 1998.