A WEB DISTRIBUTED INCIDENT AND HAZARD MAPPING SOLUTION FOR LIFE SAVING VICTORIA: AN OPEN SOURCE SOFTWARE CASE STUDY

NICOLA WALDRON[1], NICHOLA GARNETT[2], ANGUS MCAULAY[3], SHOAIB BURQ[3], SIMON FLANNERY[3], & MARCUS RESTON[3]

[1]Life Saving Victoria, 43, Dalgety Street, Oakleigh, Victoria 3166
Tel.: +61 3 9567 0000 Fax.: +61 3 9568 5988
Email: nicola.waldron@lifesavingvictoria.com.au

[2]Office of the Emergency Services Commissioner, Department of Justice, Victoria, GPO Box 4356, Melbourne, Victoria 3001
Tel.: +61 3 8684 7922
Email: nichola.garnett@justice.vic.gov.au

[3]Victorian Partnership for Advanced Computing, Spatial & Visualisation Technologies, 110 Victoria Street, Carlton South, Victoria 3053.
Tel.: +61 3 9925 3263 Fax.: +61 3 9925 4647
Email: gus@vpac.org, sab@vpac.org, flannery@vpac.org, marcus@vpac.org

Keywords: **LIFE SAVING, GIS, WEB, OPEN SOURCE SOFTWARE**

ABSTRACT: Volunteer Life Saving organisations play significant roles in promoting community awareness of water safety issues, and in the provision of patrols and rescue services at many beach locations.

Life Saving Victoria (LSV) has a committed business objective to enhance the capacity of its staff and volunteers to provide these services, regardless of location or computational resource.

The use of GIS provides useful decision support tools to many of these activities. This paper details the design and implementation of an Open Source Software (OSS) based web distributed Geographic Information Systems (GIS) to provide business driven GIS services to LSVs staff and members.

INTRODUCTION

LSV has noted the relevant use of GIS and Command & Control softwares for decision support to Emergency Services Organisations (ESO), but have until recently had limited operational use of such systems.

The current spatial information capabilities of LSV are primarily focused on supporting the activities of the Coastal Risk Management Department. The use of Global Positioning Systems (GPS) and pocket PC devices for GIS data acquisition is well established, and the subsequent integration of these data into LSVs enterprise GIS for coastal audit and hazard identification mapping purposes has demonstrated the value spatial systems for decision making.

A strategic requirement was identified by LSV to investigate further potential uses of spatial information within the wider organisation for enhanced decision support. Given the nature of operational life saving being “real time”, a key focus of this requirement is in the investigation of real time spatial systems.

Commercial vendors offer a wide range of software products to fulfil these service requirements, however, these software products typically attract significant purchase and maintenance costs.

The investigation of the use of OSS modules to develop an enterprise GIS software solution was a principal investigative process for this project.

Additionally, in line with other affiliated national organisations, a strong focus has been adopted for the development of information technology initiatives to improve the availability of information to support operational surf lifesaving, training programs and public education.

The available “club-house” infrastructure may in the worst case be considered to be heterogeneous, potentially obsolete residential quality compute infrastructure with dial-up Internet service. The lack of standardised and contemporary Information Technology (IT) infrastructure at life saving clubs outside of LSVs corporate office would thus impede the rollout of “whole of organisation” desktop IT initiatives without infrastructure upgrades.

A logical spatial information system to operate within these environments is a standards compliant small footprint web distributed mapping application to access a resource intensive server-side software application.

This paper outlines the core functionality of the application developed under these constraints by participants of the Victorian Partnership for Advanced Computing (VPAC) “Summer Intern Program”. VPAC is an independent “not for profit” Advanced/High Performance Computing (AC, HPC) service provider established in 2000 by a consortium of Victorian Member Universities. VPAC is a founding partner of the Australian Partnership for Advanced Computing (APAC), which provides Victorian access to the National supercomputing facility located at the Australian National University, and access to other APAC programs.

PROJECT OBJECTIVES

The objective of this project was to develop a "proof of concept" Internet distributed mapping application to demonstrate the value of the collection and reporting of spatial information to LSV. This may be expressed in functional real terms as:

"The development of mapping and GIS functionality to support LSV staff and volunteers in better decision making."

A key functional objective of the project was that the application needed to provide the capacity to allow information to be robustly collected into a server-side repository and reported along with other information to users in a palatable format, regardless of their location.

Additionally, the application required user interaction via a simple Graphical User Interface (GUI) given the varying levels of familiarity with GIS software within the LSV user base.

Components of this functional objective may be summarised as:

1. Providing lifeguards and volunteer lifesavers with access to information including the location of hazards, facilities, historic incident information and current weather conditions.
2. Providing tools to allow spatialised online data entry of key elements of LSVs operations including: Incidents, hazards, and patrolled beach flag locations.

Given the budgetary constraints experienced by LSV, a very real supplementary objective was to investigate the applicability of utilising Open Source Software for the development of a solution that could be deployed using minimal infrastructure upgrades.

REQUIREMENTS ANALYSIS

Given the limited availability of development resources for this project (VPAC Internships run for a period of 12 weeks), it was necessary to ensure optimal use of those resources in the delivery of the required software solution.

A preliminary analysis of LSVs requirements was therefore conducted to provide a categorical description of the required software functionality. The results of this analysis were incorporated into a project scoping document; this document also included a concise functional specification. This document proved to be most useful in providing a framework for the software development plan, which in turn was used extensively to monitor progress and identify potential problematic scheduling issues. The scoping document also provided a post-development checklist which was effectively used to assess the success of the software development exercise.

SOFTWARE FUNCTIONAL REQUIREMENTS

Core elements of the required functionality are as follows:

Web Application Functional Requirements

- Authentication and three tier level of authorisation (Local, State and National Level).
- Basic GIS tools: Pan, Zoom, refresh, view-history, full extents
- Cartographic features: legend, scale, coordinates tracking
- Redlining tools for drawing point and polygons

Read-Only Data Integration Requirements

- Base layers: Blue Marble (Australia), Landsat Mosaic (Victoria), Background GIS Layers from Geoscience Australia,
- Coastal Features: Club location, EBAN & ABSAMP databases[1]
- Live Weather feeds from Bureau of Meteorology (BOM)

Transactional Data (redlining)

- Beach Hazards (Hazard defined by a general region: Temporal in nature: permanent versus transient, display based on time range)
- Structures (Point location of typical coastal structures such as piers, jetties)
- Rescues (point location of a rescue)
- Flags

Search by club name and quick links to club locations.

SERVICE ARCHITECTURE

Overview

A three tiered software architecture comprising of **Web Application, Web Service** and a **Spatialised Database** (figure 1.) was selected to provide the required functionality.

The **Web Application** was built using MapBuilder, a web mapping client that is compliant with Open Geospatial Consortium (OGC) standards. MapBuilder allows the display of maps from a variety of sources published by OGC Web Map Service (WMS) or OGC Web Feature Service (WFS) services. WMS allows the display of maps delivered as rendered (raster) images, while WFS allows the streaming of geographic elements or features which may be temporally updated.

It is of principal significance to note that MapBuilder is deployed on the client (user) computer using resources available in any typical commodity computing environment (web browser plus internet connection).

[1] See Glossary for abbreviation definitions

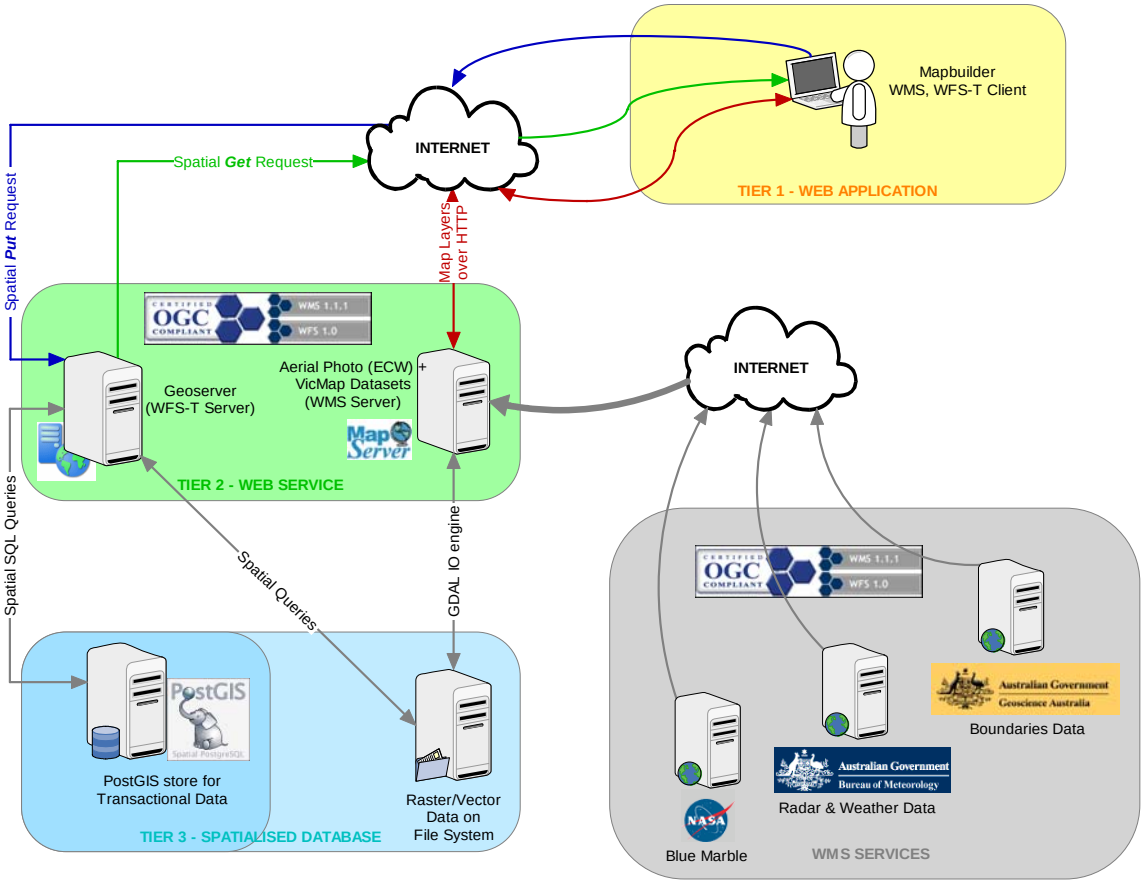

Figure 1. Diagrammatic Representation of Three-Tiered Software Architecture.

The next tier in the model, the **Web Service**, was deployed on a server-side Linux distribution. Robust Web Server functionality was implemented using Apache, with supplementary applications including MapServer CGI (WMS) and GeoServer (WFS) and transactional WFS (WFS-t) to provide the server streams for the client web application. MapServer was used to service all raster data map requests providing the WMS services, while GeoServer provided all the WFS and WFS-t capabilities, and connection to the spatialised database. A J2EE environment, Tomcat, was required to support GeoServer.

The third tier in the architecture is the **Spatialised Database**. PostgreSQL was selected for this component, and stores the transactional and persistent entities of the service. PostgresSQL was selected due to its capability to store spatial information using PostGIS extensions, and was also deployed on a server-side Linux platform.

All software components selected for inclusion are Open Source and OGC standards compliant, ensuring that the application would remain royalty free for its current deployment and be forward-extensible for future development.

The Value and Use of Open Standards

Since the advent of web browsers and associated static web content in the early 1990's, there has been much expectation regarding the role of Web based applications functioning as discrete and distributed software applications on the user desktop.

It is only comparatively recently however that modern web browsers have addressed this collective expectation. The integration of **A**synchronous **J**avaScript **A**nd **X**ML (AJAX) into web browsers allows web pages to behave like typical desktop applications without having to refresh the whole page.

Due to the number of GIS functional software components that were required to communicate with each other, easy transfer of information between software components was critical for the applications functionality and stability. Intercommunication of the assorted software components was possible by utilisation of the OGC Geographic Markup Language (GML) standard. The majority of the web service was implemented using just two specifications; the Web Mapping Service (WMS) open standard to request maps, and the (transactional) Web Feature Service (WFS & WFS-t) open standards to extract and insert map features. Furthermore, the use of GML SOAP allowed robust communications between all components within the AJAX framework.

Web Application

MapBuilder is a client-side JavaScript library of mapping widgets and other GIS tools that can be embedded into a web page transforming it into a web application that offers many functional components of desktop GIS softwares.

The design and implementation of the Web Application with MapBuilder used the model-view-controller (MVC) design pattern. This was selected because it provided separation between the content, the presentation of the content, and the application state (Buschmann et al 1996). The MapBuilder project had already adopted this design pattern, and furthermore already used AJAX technologies for both GML communication, and user interface event handling and map rendering. This close alignment of software design with the selected development environment facilitated rapid and comparatively simple application development.

The MapBuilder widgets used in this project included the main map pane, title, legend and scale, and a widget toolset that allowed the user to pan, zoom and refresh the main map. MapBuilder was also used to track the user's map navigational history and provided the ability to revisit past map locations via simple "back" and "next" buttons.

For ease of use, it was important to get the web application to function like a typical desktop application, featuring low transactional latency, very few (or no) full page refreshes, while continually servicing the user's request from the server. Historically this has been a major limitation of web applications. The use of AJAX allows data to be loaded in background from a server, offering an application environment that does not require user interaction for refreshing. Since MapBuilder has already incorporated AJAX technologies, this feature set was implemented with little supplementary effort.

MapBuilder was additionally setup to provide the functionality to redline points, lines and polygons straight onto the main map pane, allow the user to populate specific attributes of the feature before committing the redlined feature and attributes to the

back-end GIS database. Features may be inserted, deleted if entered in error, and queried for reporting purposes.

MapBuilders widgets and icons were customised for visual appeal and compliance with guidelines provided by LSV regarding user expectation. For example, due to the large number of map sources and the resultant legend, the legend widget was visually redesigned, to form a collapsible and expandable tree to conserve valuable screen real estate. Also, the attribute fields of each of the features were restricted by using a simple dropdown list and the timestamp of the feature was set server-side. These simple measures helped to increase the accuracy and reliability of the data store on the GIS web server.

The GIS Web Service

The middle tier of the system was based on two GIS Web Services, MapServer and GeoServer.

The functionality of both MapServer and GeoServer is provided through open standards specified by the OGC. The use of these open standards is fundamental to the flexibility of the developed software solution, since it allows for the integration of a diverse range of modular software components.

The GIS Web Service - MapServer

MapServer is a mature Open Source GIS Web Service framework. It makes use of the Geospatial Data Abstraction Library (GDAL) and OGR[2] libraries to provide web enabled GIS functionality, and can use both GIS format files and ODBC database connections. MapServer utilises the OGC Web Mapping Service (WMS) standard to compile GIS data into maps rendered server-side into an image which is relayed to the client application.

In this project, significant volumes of satellite and aerial photographic data were streamed as WMS services (OGC, 2006). In order to minimise the number of selectable layers and thus simplify the browsing process for the user, pre-defined cascading resolutions of imagery were used to display the most appropriate resolution of imagery based on the map's viewing scale (figures 2 & 3).

The use of open standards means that external data sources can also be incorporated into applications. The Australian Government Bureau of Meteorology provides both Web Map services and Web Feature Services for Meteorological and Oceanographic data.

2 See http://gdal.maptools.org/ogr/ for an expanded explanation of the meaning of this acronym

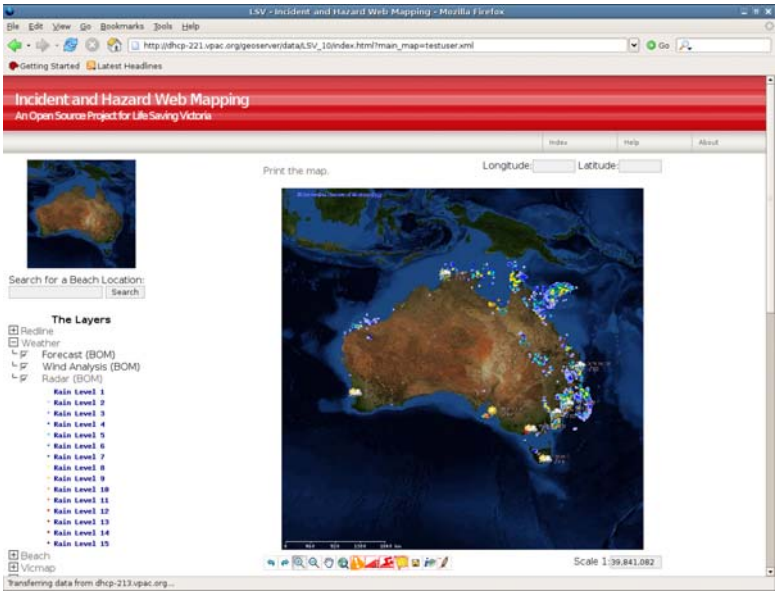

Figure 2. Screenshot of LSV Incident and Hazard Mapping Application. Continental scale NASA Blue Marble satellite image overlain with Bureau of Meteorology radar precipitation and forecast data.

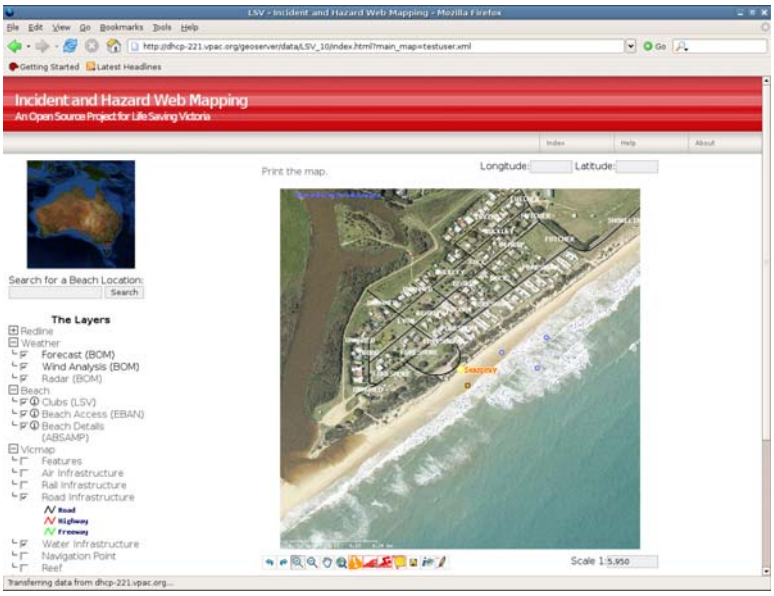

Figure 3. Screenshot of LSV Incident and Hazard Mapping Application. Local area zoom of the Seaspray Life Saving club aerial orthophotomosaiic overlain with infrastructure, incident and hazard data. A centroid of the club location is also shown.

Several of these data streams were incorporated into the application, including radar derived precipitation and wind direction and speed analysis (figure 2). For more information regarding BOM WMS and WFS services, refer to the following URL: http://ows.bom.gov.au/mapserver/

GIS Web Service - GeoServer

GeoServer is a newer framework based on the GeoTools libraries and can also access both GIS format files and database entries. GeoServer is intended to only support

vector data and offers additional functionality not supported by MapServer. Principally of relevance to this project are the transactional operations which allow data to be modified by client-side applications, and is used in the application to provide map layers which can be edited via the web interface, such as rips, hazards and rescues.

Another notable point about GeoServer is its implementation of **`LockFeature`** and **`GetFeatureWithLock`**, as defined by the WFS specification (OGC 2005) that prevents the modification of a feature by more than one user, thus ensuring the integrity of the data when it is being modified by the client.

GIS Database

The base tier of the application is a data store consisting of GIS format files and a spatially enabled database. For data which are not modified or queried, the method of storage with optimal retrieve and display performance is native GIS or Image Processing software format files. This was nominated as the preferred storage strategy for raster data including satellite imagery and aerial photomosaiics.

For vector and tabular data which are queried and possibly modified it was considered preferable to store these data in the database, since they offer indexing and rollback which improves performance, reliability and data integrity.

Spatial databases are an extension of general purpose databases; they provide spatial indexing and support spatial queries. These features improve performance for geospatial applications. For this system the open source database PostgreSQL was used with the PostGIS extension to provide spatial features. PostGIS implements another OGC standard (Simple Features Specification for SQL) for spatial extensions to the SQL language, and is widely supported by other GIS softwares. PostGIS also allows the creation and use of R-Tree spatial indices (Guttman 1984) based on the GiST indexing method inherent in PostgreSQL. This can provide significant performance gains when making spatial queries.

TRANSACTIONAL DATA COLLECTION

The implementation of web based transactional spatialised data collection offers significant benefits to dispersed organisations such as LSV, since it facilitates the consolidation of data from many locations in a standardised format. The end objective of such an exercise is to improve the quality and reduce the latency of delivery of information relevant to decision support.

Due to the significance of this feature to the business objectives of the application, a process flow diagram of the transactional component of the application is summarised in figure 4, using the example case of inserting the location of a “safe to swim” flag.

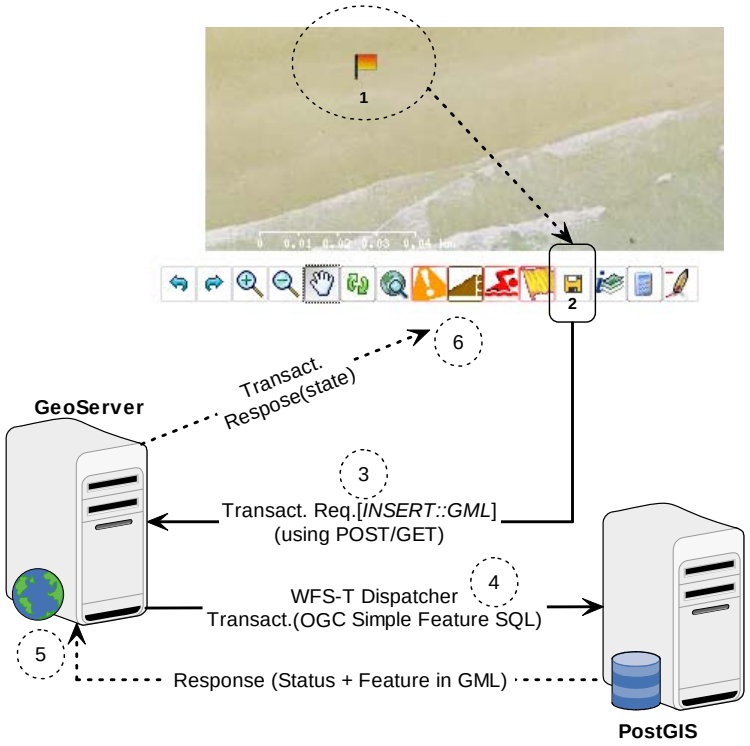

Figure 4. Representation of insertion of a beach flag into the spatial database

1. The user selects the flag tool and digitises the location of the flag using the web application tool
2. The user selects the “save” button
3. The web application sends a WFS request of the type “transaction” to GeoServer with the parameter being the GML encoded feature and attributes to GeoServer
4. GeoServer uses a WFS dispatcher which connects to the relevant PostgreSQL/PostGIS data-store that makes an OGC compliant Simple Features Specification SQL insertion to the data-store.
5. Based on the success or failure of the SQL “insert”, a status message is passed back to the dispatcher, along with the feature inserted (if successful), or error (if unsuccessful) to update the client application.
6. The user may subsequently query the flag layer to display the current location of flags at the nominated beach.

The resultant entry that such a transaction generates in the PostgreSQL database is shown in table 1.

The table contains multipoint entities with coordinates stored as latitude and longitude, with a timestamp to identify when the flag was created. Supplementary fields also stored include user identification, user association and flag persistence (permanent or temporary). It should be noted that this database can in practice be any data store such as DB2, ArcSDE.

Table 1. Database entry for beach flag

```
lifesaving=> select gid, astext(the geom), timestamp from flag poi
limit 1;
 gid |          astext                 |          timestamp
-----+----------------------------------+----------------------------
1 | MULTIPOINT(143.992187777778 -38.51762) | 2006-01-30 15:18:19.798215
(1 row)
```

CURRENT STATUS

The application has been built using contextual image data for five Victorian and one Interstate Life Saving Clubs. Club members are evaluating the application for usability, functionality, and compliance to core business objectives of life saving activities.

The evaluation period is scheduled to run for six months, with a review of the applications use and success at the end of this period. Following this review, an assessment of any required software modifications will be made to identify the resource requirement necessary to support a production rollout of the application if desired.

CAVEATS AND LIMITATIONS

Functionality was implemented using a number of software packages including PostgreSQL/PostGIS, GeoServer, MapServer, Tomcat, Apache, and MapBuilder. The configuration and maintenance of these numerous packages required specialist levels of knowledge.

A number of the Open Source projects utilised in this exercise are new, and feature little documentation which at times impeded rapid implementation. Additionally, by the very nature of OSS, the provision of support in resolving technical problems is largely dependent on a collective peer network.

Given the developmental status of some of the OSS projects used, frequent changes were required to support newly developed functionality. Close adherence to software development “best practice” of version control was therefore necessary to support and test multiple versions of the application encompassing multiple versions of component software.

Although not a technical limitation, the user defined functional specifications of the project stretched the limits of configurable software functionality, and in some cases required the development of supplementary software modules to achieve required functionality.

CONCLUSIONS

The initial effort invested in defining user requirements as a functional specification, and the development of a concise project plan were considered to have been principal

elements necessary for the success of the project. This process and resultant documentation provided a mechanism for clear communication between participant organisations and ensuring focus of the project scope.

The software platform successfully satisfied the functional specifications within the allocated human resource budget. It is of note to emphasise that the delivered application will function on heterogeneous residential quality computers supported by a standard 56k dial-up Internet connection. From a software development perspective, it is therefore considered to have been a successful project.

The technical success of the project has demonstrated that a transactional spatial information system can be implemented using Open Source software packages. These systems may be considered as a serious alternative to commercial software packages provided that a careful reconciliation of required versus deliverable functionality is made.

Additionally, careful consideration of the technical capabilities required for the development and ongoing support of these systems is also necessary to ensure a successful deployment.

Technical issues aside, the major measure of the success of this application will come from the evaluation process, and from the uptake of the application by operational life saving staff and volunteers in summer 2006-2007.

REFERENCES

Guttman, A., R-Trees: A Dynamic Index Structure for Spatial Searching, *Proc. ACM SIGMOD International Conference on Management of Data*, April 1984.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., (1996). *Pattern-Oriented Software Architecture.* John Wiley and Sons.

Open Geospatial Consortium Web Feature Service (WFS) *Implementation Specification*, Version 1.1.0, 2005-May-03 04-094

Open Geospatial Consortium Web Map Service (WMS) *Implementation Specification*, Version 1.3.0, 2006-Mar-15 06-042

GLOSSARY

ABSAMP	Australian Beach Safety and Management Program
AJAX	Asynchronous JavaScript and XML
APAC	Australian Partnership of Advanced Computing
Apache	Apache HTTP Server is a free software/open source HTTP web server
Apache Tomcat	Apache Tomcat is a web container developed at the Apache Software Foundation
BOM	Bureau of Meteorology
CGI	Common Gateway Interface
EBAN	Emergency Beach Access Number
GDAL	Geospatial Data Abstraction Library
GIS	Geographic Information Systems
GiST	Generalized Search Tree
GML	Geography Markup Language
GUI	Graphical User Interface
HPC	High Performance Computing
HTTP	HyperText Transfer Protocol
IT	Information Technology
Java	An object-oriented programming language developed by Sun Microsystems
JavaScript	A scripting programming language based on the concept of prototypes
J2EE	Java 2 Platform, Enterprise Edition
LSV	Life Saving Victoria
MVC	Model View Controller
ODBC	Open DataBase Connectivity
OESC	Office of Emergency Services Commissioner
OGC	Open GIS Consortium
OGR	See http://gdal.maptools.org/ogr/
PHP[a]	Personal Home Page (the original name for PHP server side scripting language) an embedded scripting language
PHP[b]	recursive name for Hypertext Pre-processor
R&D	Research & Development
RRA	Registered Research Agency
R-Tree	R-trees are tree data structures that are used for indexing multi-dimensional information
SOAP	Simple Object Access Protocol
VPAC	Victorian Partnership of Advanced Computing
WMS	Web Map Service
WFS	Web Feature Service
WFS-t	Web Feature Service (transactional)
XML	Extensible Mark-Up Language