EchoBeach

A system for generating blogs from dive logs

A third year software engineering project at the University of Aberdeen
2005 – 2006

Group Report

Ian J Robotham
Christopher JA Cowan
Euan TD Campbell
Leanne Eddie
Hartnell Foster
Lanlan Wang
Contents

Contents ......................................................................................................................... 2
Introduction .................................................................................................................... 4
Overview of project ........................................................................................................ 4
Summary of Team Organization ..................................................................................... 4
   Figure 1. Team Structure ......................................................................................... 5
Project management strategy ....................................................................................... 5
Requirements for Proposed System ............................................................................. 6
Project Working Name ................................................................................................. 6
Functional Requirements ............................................................................................. 6
   Figure 2. System Overview ..................................................................................... 6
Top-level Requirements ............................................................................................... 6
Specific requirements ................................................................................................... 6
Performance Requirements ....................................................................................... 8
Logical Database Requirements .................................................................................. 8
Design Constraints .................................................................................................... 8
Quality Attributes ....................................................................................................... 9
Feasibility Study ......................................................................................................... 10
J2SE[6] ....................................................................................................................... 10
PHP[7] ....................................................................................................................... 10
HTML[4] .................................................................................................................... 10
MySQL[8] .................................................................................................................. 10
Organisation ................................................................................................................ 10
Product Specific Risks ............................................................................................... 11
Generic Risks .............................................................................................................. 11
Additional Requirements .......................................................................................... 12
Design .......................................................................................................................... 13
Overview ..................................................................................................................... 13
   Figure 3. Activity Diagram for Blogging Engine ....................................................... 13
   Figure 4. Activity Diagram for Client ..................................................................... 14
   Figure 5. Sequence Diagram .................................................................................. 15
System Architecture .................................................................................................... 16
   Figure 6. System Architecture .............................................................................. 16
Software Architecture ................................................................................................ 16
   Figure 7. Blogging Engine Architecture ................................................................. 17
   Figure 8. Client Architecture ................................................................................ 17
System Interface Description ..................................................................................... 17
Data Management ....................................................................................................... 18
   Figure 9. Entity Relationship Diagram ................................................................ 18
User Interface Design ................................................................................................. 19
Client ........................................................................................................................... 19
Sample Screen Layouts for the Client ...................................................................... 19
   Figure 10. Java Client Example ............................................................................ 19
   Figure 11. Java Client Example 2 ......................................................................... 20
   Figure 12. Java Client Example 3 ......................................................................... 20
Blogging Engine ......................................................................................................... 20
   Figure 13. Echo Beach Homepage ....................................................................... 21
   Figure 14. Great Barrier Reef dive blog ............................................................... 21
   Figure 15. Comments page ................................................................................... 22
Design Changes ......................................................................................................... 22

Cont.
Cycling and Integration
Examples of our Coding Standards in Operation
Figure 16. Coding Standards Example
Key Implementation Decisions
Figure 17. Demonstration code reusability
Figure 18. Code Abstraction
Testing
Testing Strategy
Test Results
References
User Manual
Registration
Novice User
Importing your dive
Open a dive
Save a dive
Personal Details
Preferences
Maintenance Manual
Installation
Compilation
External Dependencies
Source Code - Client
Source Code – Blogging Engine
Introduction

Overview of project

The core purpose of our proposed system is to create blog (Internet based diary) using data captured from a dive computer. The concept is a diver will wear his or her dive computer while on a dive which will gather information about the dive, for example, the depth the diver reached, the duration of the dive, the air and water temperature oxygen use, GPS locations etc. Following the dive, the diver can connect his dive computer to his PC and then import CSV data into our system. The information captured by the dive computer will then be utilized by our system to create a blog which can be permanently published to the Internet. The blog will clearly show all the vital statistics of the dive plus have a section to view pictures and video footage of the dive.

Our target market will be the diving community as a whole, we hope to sell to everybody from amateurs just starting out and looking for a way to share their diving experiences with friends and family, right up to professional diving schools who can create online portfolios of dives and a community aspect to their business (with the potential for them to offer additional services).

We expect the impact of our software to be substantial, by offering the ability to generate blogs from dive computer data, we are offering a feature that many of our competitors are not. The fact that our competitors are not offering this feature means we will be bringing something unique to the market and giving customers another serious option when investing in diving equipment.

The project can be divided into two sections, a client and a blogging engine. The client provides a graphical interface for the user to upload and process data to the blogging engine which generates images on the fly and displays the information.

Summary of Team Organization

The team (team echo) was formed on 30th September with six members: Euan Campbell, Christopher Cowan, Leanne Eddie, Hartnell Foster, Ian Robotham, and Lanlan Wang.

Throughout the whole project the team has been organised to have at least two team members working on each task, to enable continuity in the event of sickness or absence of a team member. This allows the team to take proactive measures to avoid project overrun.

Team Echo has been organised into the hierarchical structure shown in figure 1.
As the course required, we have a team website (http://www.csd.abdn.ac.uk/~teame/teamecho/), which contains functionality similar to Sourceforge.net(1). Team members can download files from the documents & resource section and upload their own files. There is a bug tracker for members to post bugs and a history of action taken can be recorded alongside this.

During the first term the team had two meetings each week held on a Monday and a Friday. In Monday meetings, the team leader gave out the tasks to each subgroup, and then during Friday meetings, which were held with the project guide, feedback was given on group progress.

During the second term the team had one meeting each week on Tuesdays held with team members and the project guide, members gave feedback on the tasks which were allocated in the previous week, then the team leader gave new assignments.

Minutes of Meetings were taken by Hartnell Forster and uploaded to the team website.

The whole system was completed for just after the Easter break with the whole of the Blogging Engine completed by mid-March and the Client finalised over Easter. Test plans were generated prior to Easter and partially ran before Easter and over the Easter break, well ahead of the project deadline.
Requirements for Proposed System

**Project Working Name**
The working name for the project is “Echo Beach” but could be changed for the release production.

**Functional Requirements**

![Echo Beach System Overview](image)

**Top-level Requirements**
Figure 1 shows the top-level requirements of our system; functions can be divided into 2 domains:
- **Client**
- **Blogging Engine**

**Specific requirements**
Upload/Store Data:
- user has to upload data from the dive computer into Echo Beach.
- mandatory function of the system.
- the actor is the user.
- pre-condition: user installed the software onto their PC's.
- post-condition: data will be stored into the system database.
Analyse/Show Dive Log
- mandatory function of the system.
- pre-condition: the user has uploaded data.
- post-condition: the dive data is generated.

Modify Dive Log
- this shows how to add media, the dive diary and buddies into the dive log. The updated information is then changed to HTML.
- optional function of the system.
- pre-condition: the user has uploaded data.
- post-condition: the existing dive log is updated with the new details.

Publish Blogging page
- this publishes the dive log as a Blog using the PHP script.
- optional function of the system.
- pre-condition: the dive log must be formatted as a HTML/PHP document.
- post-condition: dive log published on-line as a Blog.

Search Blog
- enables users to search the database using a search phrase e.g. for a diver's name, for dive location.
- actors can be either registered or unregistered.
- Optional function of the system.
- pre-condition : dive log has been published on-line.
- post-condition: users can view the public Blogs.

Modify Blog
- this shows how to add/delete existing media, dive diary entries and buddies from the selected dive log.
- optional function of the system.
- pre-condition: the user has uploaded a Blog.
- post-condition: the existing dive Blog is updated with the new details.
  The web is updated.

Add Comment
- this shows how to add a comment to a Blog and change format of the Blog.
- actors can be either registered or unregistered.
- optional function of the system.
- pre-condition: accessing the dive log which the user wants to comment about.
- post-condition: comment has been added and Blog has been updated.

Convert to HTML/PHP
- generates XHTML formatted code.
- pre-condition: the user has uploaded the data from their dive computers.
- post-condition: the data is converted to HTML/PHP.
Add/Edit Dive Diary
- add a new dive diary to one of your dives or edit an existing one.
  - pre-condition: the user has uploaded or will shortly upload a dive log to the web.
  - post-condition: a new dive diary is added, or an existing one is edited and updated.

Add/Edit Media
- add media to one of your dives or add more/delete media from an existing one.
  - pre-condition: the user has uploaded or will shortly upload a dive log to the web.
  - post-condition: media is added, or an existing one is edited and updated.

Add/Edit Dive Buddies
- add dive buddies to one of your dives or edit/delete existing ones.
  - pre-condition: the user has uploaded or will shortly upload a dive log to the web.
  - post-condition: dive buddies are added, or existing ones are edited and updated.

Delete Blog
- delete an existing Blog from the web
  - pre-condition: the user has uploaded a Blog and now wants to delete it.
  - post-condition: the Blog is deleted from the web and removed from the database.

**Performance Requirements**

The system will respond within five hundred milliseconds to user input. The system shall take approximately thirty seconds to upload to the remote database with an additional approximate thirty seconds for each multimedia item. Progress feedback will be presented to the user (in the form of a progress bar, animation or other progress feedback widget).

**Logical Database Requirements**

We will use a RDBMS which supports the ANSI SQL standard(2). Only authorised users will be able to gain access to the Database. The tables created are Dive Index (the bulk of information about dives), Dive Table (information related to a specific dive), Diver Details (information about the diver), Location (information on a specific location), Media (entails pictures, video and audio which can be included using our blogging engine), Registered Users (information about registered Users (divers) including User details and password). Primary and foreign keys have been selected through the normalisation process.

**Design Constraints**

**Standard compliance**

IEEE standards(3) will be used for documentation and coding where applicable. The Team Echo Coding Standards (available through the team portal) will be used to ensure quality code generation.
Language
The system should run on multiple platforms. Thus the client should be written in a platform independent language. The nature of the web is platform independent thus any page written in standard HTML/XHTML(4) will comply. Communication with the database should be written in ANSI standard SQL(2).

Quality Attributes
Availability
The Blogging Engine (the live Blogs) will be available to several categories of user: general visitor, blogger (somebody who has installed & uses the client), moderator (for use in community situations when the system administrator requires assistance) and the system administrator. Each of these will access to different features and areas of the Blogging Engine. The client will be available to any registered user who has installed it.

Security
User names and passwords are required to log on to the system and identify the level of access the user has to system resources. Personal passwords will be encrypted using the MD5 hashing algorithm(5).

Maintainability
The system administrator will have authorization to perform routine maintenance on the system. Comprehensive code commenting, documentation and conformity to standard coding practices will enable programmers to make structural changes to database, GUI or processing components.

Portability
The system will be written in a platform independent language, enabling it to work efficiently on Unix, Linux, Sun Solaris, Apple MacOS and Microsoft Windows based operating systems.

Robustness
In accordance with IEEE standards, the system should have 95% uptime, notwithstanding operating system or hardware failure. Validation of user input will prevent errors.

Learnability
The system will be menu-driven combined with other graphical user interface elements. Every screen will be of the same format to reduce learning time; furthermore wizards will simplify the learning process, enabling the user to generate blogs rapidly and easily. A User Manual will be provided with the system alongside an integrated help facility.

Configuration
The User will be given a series of options to change settings including alternative User Interfaces, Database Connectivity settings and locale-specific formatting. The whole system will support internationalisation.

Packaging code
The code for our Blogging Engine will have a modular design so that it can be reused for other projects. The client will implement a Wizard interface and configuration functionality both of which shall be available for re-use.

Installer
For our installer, Team Echo will use a standard installation wizard created using one of many freely available installation tools.
Feasibility Study

Technologies

**J2SE(6)**
- Platform- independent Object Oriented language with built-in provisions for GUI design (Swing) and database access (JDBC). Extendible with custom packages.
- High level language – runs slower than other Object Oriented languages (such as C++).
- Freely available. IDEs such as Eclipse and NetBeans freely available online. Educational licenses available for students and institutions wishing to use IDEs such as Borland JBuilder or Together.
- Individually, team members have approximately 2 years’ experience (some may have slightly more) with J2SE and Object Oriented programming techniques.

**PHP(7)**
- Standard scripting language for Web development. Versatile, with built-in support for database access.
- New releases may not support functions native to older versions of PHP.
- Freely available online.
- Generally limited team experience.

**HTML(4)**
- Standard language for Web development. Universal, recognised by all browsers available.
- Older browsers may be incompatible with graphical elements.
- Generally limited experience with HTML/XHTML.

**MySQL(8)**
- Supported by most server platforms and compatible with technologies such as PHP and J2SE.
- Licensing, though clearly stated in documentation, is unclear in practise.
- Freely available for non-profit purposes. Again, licensing not totally clear in practise.
- Generally limited experience with HTML/XHTML.

Strengths, Weaknesses, Costs and Team Domain Experience

**Team**
The team consists of six Junior Honours Undergraduate students. Despite common experiences, members have varying skill levels with the technologies used in this project. On the evidence of previous years, it is considered a genuine concern for project feasibility that the team may not survive as a unit. There is always a very real possibility that members could leave the team or university, or that apathy amongst some team members could cause setbacks in the project schedule.

**Organisation**
Although the overall structure is hierarchical and may seem very autocratic, each of the team members has been assigned an area of responsibility, for which they will assume a leadership role. This enables all team members to experience management and leadership. The schedule identifies each person’s activities and highlights responsibilities, and is made available for all to see.
Competition
The main competitors in this area are SharkPoint(9) and Internet Dive Log(10). Each offers slightly different capabilities – SharkPoint creates electronic dive logs from dive computer data, allows synchronisation between desktop and portable devices, can include GPS data, and is compatible with Windows, Pocket PC and Palm operating systems, but is not web-enabled; Internet Dive Log also creates electronic dive logs from dive computer data, and is web-enabled, but only through its publisher's website. Our proposed system should offer Blog creation from dive computer data, inclusion of GPS data, and be web-enabled by way of a Blogging Engine deployable on all compatible web server platforms. The Blogging Engine should be skinnable, configurable for language and locale, and also permit multimedia inclusion and buddy-linking (hypertext linking between diver Blogs) for particular dives. Several other freeware dive logging software exists, but quality is not always assured.

Risk Analysis & Management

Product Specific Risks
The system will have to run on all platforms, and so it may be time consuming to implement this. However we are confident that with the abilities we have in the team, we will be able to overcome this problem. The languages we are using to code our system will be platform independent, this should make this problem much easier to deal with.

We conducted an interview with a diving school company(11) in Aberdeen and when asked about their views on how potentially successful they think our system could be, they highlighted the fact that it may be looked upon by divers as an online logbook. The problem with this being that most divers see filling in the logbook as a chore rather than something they enjoy doing. This could be seen as a major risk to the fundamental idea behind the project. However, our system is designed as more of a place to share your diving experiences with friends and family in a sociable environment rather than a logbook which you have to spend time writing up.

Generic Risks
This type of task is new to all of us, and because we have a lack of basic experience in the skills needed for a task such as this, there may be increased pressure to complete tasks when deadlines approach.

Some team members are not as familiar with programming languages and technologies as others, so we will need appropriate management to make sure people are working towards their strengths.

Time management will be a major concern with this project. All the team members have other important commitments outside the group and this will be factor on whether or not we meet project deadlines.
**Additional Requirements**

**Hardware Interfaces**
An Internet connection will be essential for this system to function, otherwise the client will be unable to upload the data to the Internet, and similarly the Blogging Engine will be unable to function.

**External Interfaces**
The Blogging Engine requires a web server and the whole system requires a RDBMS (which may be the same physical server). This server must be capable of handling the total web traffic and data uploads and as such needs to be a reasonably powerful machine.

**Hardware and Software Requirements**
The hardware must be a computer system capable of efficiently running a platform independent language e.g. J2SE(6) or Microsoft .net(12), utilising any platform, and able to connect to the Internet. The Server system must be capable of handling all the requests passed to it and must efficiently run a web sever and RDMS under any platform.

**Communications Interfaces**
The communication protocols used will be HTTP(13) and some form of standardised database connectivity e.g. ODBC(14), JDBC(15), ADO(16).

**Memory Constraints**
There are no specific memory constraints.

**Site Requirements**
The system should be fully configurable for different environments (such as utilisation of various RDBMSs and support Internationalisation).
Design

Overview
In order to choose our final design Team Echo met up on various occasions to compare drawings and share ideas. From this we were able to choose, what we would describe as, the best overall design. Using teamwork meant that a variety of different ideas could be looked at and the better ones selected for use within the project.

After talking a bit about our project we realised that for completion we would require two separate interfaces. One would be the Client and the other the Blogging Engine. The Client would be the software on the users computer which they can use to upload their dive details. The blogging engine is the Internet component of the software. The blogging engine provides a common ground where people can browse through all the uploaded dive logs.

- For users who don’t need the following actions:
  - ClickToConvertToHTML

  - ChooseToPublishDiveLog
    - action not allowed if it’s an unregistered user
    - PublishToOtherSite

  - PublishToWeb
    - user not allowed, back to their blog

  - ViewBlog
    - EditDiveDiary
    - DeleteBlog
    - EditProfile

  - AddComment

Figure 3. Activity Diagram for Blogging Engine

- For users who want to publish their dive log on the Internet, the system provides exit options before ChooseToPublishDiveLog action so that user can exit system.
  - Only Registered Users have the access to modify their Blog.
  - Both Registered Users and Viewers have access to view any public Blog on Echo Beach.
Figures 3 and 4 indicate potential activity diagrams for using the EchoBeach software.

- The system has to successfully receive data from dive computer otherwise an error may occur – the user has to check dive computer and return to the first action.
- The system has Simple & Complex versions for user to select
- Once the system has finished analysis, the Blog can be modified by the user to add diary entries, pictures, videos and Buddy links.
- The user could exit the system by any action (eg after updating the Blog or viewing the Blog).
Figure 5 shows a sequence diagram for the EchoBeach software system.

Figure 5. Sequence Diagram
System Architecture
The data in our system will be managed in a MySQL(8) Database because this will allow multi-user access and easy storage of large amounts of data.

The system will be written as two separate components. These are:
- Upload System (Client) – a single component to upload the data from the users computer to the database.
- Viewing System (Blogging Engine) – comprising of a couple of systems – a PHP enabled web server to retrieve and process (convert to Web Content) the information in the database and a Web Browser to display the pages to the end user.

Both the components will link to the central RDBMS.

This is demonstrated below:

![System Architecture Diagram]

Software Architecture
As mentioned above Echo Beach exists as two separate systems – the Client and the Blogging Engine. As shown in figure 2 below, the Blogging Engine consists of 3 distinct subsystems – Blog Viewing, Blog Management and System Management, each of which is available to different people.

The Blog Viewing element is available to everybody whilst Blog Management only to registered bloggers and System Management to the Site Administrators. These all use a common set of features – namely Presentation, Data Processing and Data Retrieval. Thus the Blogging Engine uses a common three-tiered layer architecture, accessed by each of the Engine components which will access slightly different elements of each layer.

Because each of the components has slightly different processing and presentation requirements, but also requires common elements, the system will be implemented using a series of common elements, which are extended (as new elements) to fulfill the system requirements. The data retrieval layer uses common elements throughout, which will be manipulated by the processing layer, thus only one data retrieval component is required.
The Client system will again utilise a model-view-controller (MVC) Architecture featuring User Interface (view), Data representation (model) and Data handling (controller). As shown in figure 5, there is a very strict control flow which the implementation (using object-orientated techniques) will enforce.

The software was designed this way to reduce the load on the server and allows the generation of pages that can be used on other systems, meaning there is no obligation to use our blogging engine.

**System Interface Description**

The system will have several interfaces:

- Blog Engine (Viewing) – interface for general Internet users viewing blogs
- Blog Engine (Blog Management) – interface for registered bloggers to manipulate settings for their blog
- Blog Engine (System Management) – interface for site administrators to control user access, delete blogs, make archives and other general system administration duties.
- Client – interface for user to import their dive data, add media, diary and other aspects to their blog. This will have both novice and expert modes.
**Data Management**

Based on the requirements analysis and specification for the Dive-blogging system, some assumptions and constraints are made to develop a high-level description of the data to be stored in the database.

- Each program must belong to one person with a unique Diver Name.
- Each individual dive has a dive index number included in the index table, and a separate dive profile table.
- Each dive site has a unique location number.
- A unique Media ID identifies all the media elements.
- The dive profile table has a combined primary key. Dive number and Time. This allows the creation of graphs such as depth at a certain time, or tissue saturation at a certain time on any specific dive.

Figure 9 shows the entity relationship diagram for the Echo Beach database:
**User Interface Design**

This application will comprise of two separate interfaces – one for the Blogging Engine and one for the Upload Client. There will be Novice and Expert modes for the Upload Client. The layout for all the interfaces needs to be clean and uncluttered and in the case of the Novice mode utilise a wizard. The web based elements should comply with the W3C specifications relating to web development (4) and WAI (Web Accessibility Initiative) (17) guidelines issued by the W3C.

**Client**

After using additional software to convert dive computer data to a CSV file, the user can then give the location of the CSV file to EchoBeach which will process the information. The user can then enter further information about their dive, provide a diary, link to their buddies and upload media elements (pictures and video files). The information can then be saved to the PC or published to the online Blogging Engine.

**Sample Screen Layouts for the Client**

Figures 10, 11 and 12 outline sample GUIs for the Client, showing different aspects of the system.

![Figure 10. Java Client Example](image-url)
Before a user can publish their dive log to Echo Beach Blogging Engine, they will be required to register with the website. This is a very short painless process which will provide them with members level access to the website.

Once a user has uploaded their dive they also have the option to add pictures, videos and dive buddies. A dive buddy is someone who you dive with. To add a dive buddy all you have to do is enter some basic information about them and this will be added into the database. There is no obligation for your dive buddy to join Echo Beach. Comment functionality is also provided; with this registered users can comment on dives. Non-registered users will be prompted to login or register with the site.
before they may post comments. The EchoBeach homepage is illustrated in figure 13.

![Echo Beach Homepage](image)

As mentioned above, the user will be asked to type in the GPS coordinates of the exact position of their dive. Using this information, the software will utilise the GoogleMaps™ API, which will then plot the dive position on a map.

Figure 14, below, is a screenshot of a dive blog, which took place in the Great Barrier Reef.

![Great Barrier Reef dive blog](image)
As shown above, it is possible to see all the information that is required to make a dive blog. At the bottom right hand side of the page there is a list of buddies who participated in the dive. Each buddies profile can be accessed by clicking Profile next to their name, it is also possible to search for dives completed by that diver by clicking on the diver’s name. This will generate a list of their dives.

The toolbar at the top gives you access to the dive diary (a personal report about the dive), uploaded media and any comments (figure 15) that have been left by users.

![Figure 15. Comments page](image)

**Design Changes**

During the latter phase of our development some changes were made to the design, this enabled the project to remain on schedule.

Examples of these changes include changing the blogging engine interface to be compatible with the data generated, dropping some functionality such as Undo/Redo, and removing the list of dives in the client interface.

These changes have not adversely affected the project, and in some cases have streamlined the application, meaning it is more efficient in performing its core task.
Coding and Integration

Before we set about coding anything in our system, a discussion was held and some coding standards were formed. These were followed whilst working on the project. We decided upon some common rules such as:

- Using comments
- Using JavaDoc
- Using white space effectively

With the help of these rules the whole coding process was more structured, enabled members to work on different sections and will enable future maintenance and the feeling of us all following a common code writing practice definitely contributed towards the successful completion of the code.

This is an extract from our coding standards document:

**Writing Code**

- Use comments – if something is not clear, add a comment; if you do something for a reason put a comment; comments cost nothing – use them!
- JavaDoc your work (where applicable) – especially if you are writing libraries which are going to be used by others. Investigation will be made into PHPDoc and a decision made on its usage.
- White space costs nothing – indent your code, add line breaks – this improves readability of your code no end and makes it easier to find bugs and errors.
- Ensure you consider security issues (Buffer Overflows, XSS hacking etc.)

**Code Review**

All code will be reviewed by at least one other person. You will be asked to review code and sign it off. Checking code before it is released to testing should cut out the number of serious faults and flaws in our system. It may seem like a pain but (hopefully) when the other groups are up ‘till 3am every night in the last few weeks trying to fix bugs, we’ll be tucked up in beds relaxing.

**Backups**

Backups should be made of all your work on this project.

Leanne Eddie will be made responsible for maintaining a weekly combined zip containing the work from that week.

**History**

Please add and maintain a history at the head of each code file you create. An example of coding style and history is enclosed.
Examples of our Coding Standards in Operation

```php
////////////////////////////////////////////////////////////////////////////
// Functions.php - Contains functions for each element #
// # of the site, incl. data retrieval, #
// # data upload, processing etc. #
// # Author(s): IJ Robotham; #
// # E Campbell; #
// # H Foster; #
// # Created: 24 Jan 2006 #
// # History: 3 March 2006 #
// # 19 Mar 2006, IJR; Fixed all errors #
// # #
// # Reviewed: NOT YET REVIEWED #
// # Copyright: (c) 2006 Team Echo, #
// # University of Aberdeen #
////////////////////////////////////////////////////////////////////////////

//<functionName> - <briefDescription>
//<briefDescription>
//<Requires: <variableName> - <briefDescription>
//<Returns: <briefDescription>
//Author: <authorName>
//Date: <date>
////////////////////////////////////////////////////////////////////////////

function getLocationName($locationNo)
{
    // include config & DAL files
    include("config/config.php");
    //include("database/mysql.php");

    // create a new DB connection
    $db = new DB($CFG_dbName,$CFG_dbUser,$CFG_dbPassword,$CFG_dbServer,$CFG_dbParam);
}

Figure 16. Coding Standards Example

Figure 16 is a typical example of how the files look. They start with a large commented area which displays information of the name of the file, the authors, the date the file was created, the coding history of the file, the reviewed status and the copyright information.

By starting all our files in this way it made reviewing the code much easier as the person who was reviewing the code just had to open up the file, read the top of the page and they immediately knew what the code was supposed to do. The numerous one or two line comments dotted throughout the code also proved to be invaluable when it came to maintaining and reviewing the code at a later date.

After the starting commented section, there is another commented section explaining the basic functionality of all the functions in this class. This section was especially useful for the reviewer who had not written the code and perhaps would not fully understand it otherwise. This section obviously only applied to files such as this one where all the functions followed a generic pattern with some key code changed. In other files where the functions were not similar, this section was obviously not relevant and so was not included.

When it came to backing up the files, a member of the team was selected to come in at the end of each week and send a zipped copy of all the code to the group guide Dr Sripada for him to back up on his hard
drive. We felt these proactive methods would avoid serious problems and delays in the event of code corruption.

The way we went about storing our files was very important. In our file space we had various sections that separated different parts of code from one another. Instead of having all the code for the entire project in one folder we separated it into sections for the website, the blogging engine and the client. A further section was for things like GUI design and images to be used on the website. Once again this design made the reviewing and maintaining of code much easier as both the people who had written the code and saved it to a certain folder, and those who had no involvement with generating the code whatsoever could both easily navigate to it.

**Key Implementation Decisions**

Throughout the development stage of the coding, some major decisions had to be made as to how we were going to go about overcoming some key issues.

Fairly early on we decided upon the idea of having two separate interfaces. One for beginner users who would need a wizard to guide them through the different components of the software and another for intermediate users who were comfortable using the components of the software on their own without any assistance.

Figure 17 shows a section of code that was used to implement the wizard. This was written in a modular way to allow numerous screens or panels to be created and used by the wizard thus enabling us to generate multiple wizards using the same core engine technology.

```java
/*
public class Wizard extends JFrame{

/**
 * Auto generated by Eclipse
 */
private static final long serialVersionUID = -2710827204083005196L;

//declare some global vars
private JButton btnCancel, btnPrevious, btnNext, btnFinish;
private JPanel screen;
private HashMap screens;
pwritableWindowScreen currScreen;
private int currPlace, id;
private boolean complete = false;

/**
 * Basic constructor to generate an instance of the Wizard
 * @param name the name of the wizard (appears in title bar)
 */
public Wizard(String name){
    //setup blank screen
    currScreen = new EmptyPanel();
    //set id variable (used to recognise each screen)
    id = 0;

    //setup screens
    screen = new JPanel();
    screen.setLayout(new CardLayout());

    //setup screens hashmap
    screens = new HashMap();

    //create the UI
}
```

Figure 17. Demonstration code reusability
Another decision that was made was to have all the database connection and access in the same location. A separate database class was created and everything to do with connecting to our team echo database and executing the SQL needed to manipulate our database was included. A section of code for this is shown in figure 18. This technique was used in both the PHP and Java systems.

The reason for doing this was so that instead of having the same code used to connect to the database and manipulate the tables repeated again and again throughout various files, we could have it in one place, then the files that needed to access the database could make use of this class making the code generally much more readable and efficient.

```java
public dbClass() // Constructor
{

    String url = "jdbc:odbc:echobeach";

    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection(url, "username", "password");
    }
    catch (ClassNotFoundException cnfex) {
        System.err.println("Failed to load JDBC/ODBC DriverManager.");
        cnfex.printStackTrace();
        System.exit(1);
}
```

Figure 18. Code Abstraction

The existing software we used in our coding included Borland Together(19) for the generation of entity relationship, test case and use case diagrams. Development of the client was primarily completed using Eclipse(20), with some GUI elements being generated using NetBeans(21). Dreamweaver(22) was used to generate HTML, CSS and PHP elements of the blogging engine.
Testing

Testing Strategy
For the testing phase of the project a separate team was set up, comprising primarily those who had less involvement in the development of the software, this left a small development team to fix any bugs or problems encounter by the testing team.

Test plans were drawn up to cover the whole system, including the UI design, execution time, data validation and error recovery. These plans detailed the exact steps which should be followed when carrying out a test. It was decided that due to time constraints security and code duplication elements would not be tested, however ideally both of these factors would have been tested.

The criteria for a baseline pass was that the system operates smoothly and it’s core functions operate without error. Errors such as the system allowing blank fields or icons not displaying will be tolerated for a baseline pass, however should be flagged as issues requiring attention in later, more rigorous tests. Processing time is also an important criterion to consider during testing, the system should perform within the guidelines set down in the requirements specification, and lengthy delays in processing or continuation should be regarded as criteria on which a system can fail.

A further criterion which would cause a test to be regarded as a fail, would be a failure in the basic operation, that is, the system failed to pass the data correctly amongst system objects and between the components.

In the event of a bug being discovered (that is a test failed) the details were recorded in the Bug Tracker on the team website. This was monitored by the development team who would prioritise the bugs and attempt to fix the problem, by releasing new versions of the code. These new updates were then rigorously retested to ensure that errors were not propagated to other areas following a “bug fix”.

As well as this final phase of testing, the development team operated in an iterative manner, creating small sections of the system, then testing them to ensure that they functioned correctly, this helped catch some early errors and improved the efficiency of the team. This dual approach in the testing of the system should lead to a stable system, with all major problems fixed.

Test Results
Each member of the testing team was assigned aspects to test and was issued with test plan questions. A sample test is included below.
Test Plan Results: Log in
Following the directions on the Log in test case scenario:

Q.1. Has the system allowed you to log in?
   A. Yes I have successfully be able to log in.

Q.2. How long did this process take? Was that acceptable?
   A. It took a matter of seconds. This is acceptable.

Q.3. Has the function performed as expected?
   A. Yes it took me to the Member Home page allowing me to change my password, manage my profile and log out.

Q.4. If enter wrong log in name/password, how the system behave?
   A. It displays a message telling me my username or password is incorrect and allows me to make a second attempt.

Q.5. Are there any error message pop up to warn the users?
   A. A small error message pops up informing the user that the wrong information has been entered.

Q.6. How’s the performance rate? (1-10)
   A. 10 out of 10. It performs perfectly and has no errors.
Summary

Overall the project can be deemed a success because a working, functional piece of software was delivered within the scheduled timeframe. However the delivered product does not match the initial client (& internal) specification. Whilst the majority of the functional elements were successfully included, elements, particularly in the client, had to be omitted and redesigned due to time and skills constraints. This should not be viewed negatively, however as much of this was additional, bonus functionality over and above the initial specification.

The major cause of this reduction in functionality was the amount of time available and the skills of the group. In our internal skills review, the majority of the team members rated their programming skills as Poor or Average, and as such the team was very much dependant upon a few developers to complete the project. These developers were often required to assist the less able team members and therefore unable to devote time to building the additional more advanced features. The other issue faced by the team was the amount of available time, because the project was running alongside other courses which had their own in-course assessment projects and assignments. This reduced the amount of time which team members could contribute toward the project, thus much of the development occurred in short bursts of activity, as opposed to a continuous stream of iterative development.

The system in its current state has many opportunities for future development, increasing functionality and flow.

- In it’s current state the system lacks some of the more advanced GUI features such as Undo, Redo, Context-Sensitive Help which would be nice to have in future editions.
- The client system was originally intended to include a list of dives performed by that user, and when clicked the details of these would be displayed in the system (to allow for post-publishing editing). Unfortunately time constraints meant this was dropped, this feature would be quite useful to develop.
- A large amount of spatial data (GPS coordinates) is collected by the system and it would be great (especially from a usability aspect) to include all a divers dives (or even all dives) plotted on a map, which when a plot point is click would open the blog. This is possible to create using the GoogleMaps™ API(18), however time did not permit us to develop this feature.
- Error checking in the system is rudimentary at best, and in some places non-existent. This should really be introduced to ensure a smooth and stable user experience.

These developments impact little on the core task of the project, which we have satisfied, however are additional elements of functionality which would improve the overall user experience whilst using EchoBeach.

From the point of view of the Computing Science Department, the main goal of the project was to demonstrate to the team Software
Engineering methodologies and to highlight and educate about issues within Software Engineering. The team, through this project, have learnt many useful skills and lessons about Software Engineering. These include that it is useful to plan ahead but not to rigidly adhere to the plan and to be flexible enough to renegotiate specifications and redevelop plans. It is also important to set deadlines and stick to them, there were several internal deadlines which were missed by the team, and whilst these did not delay the delivery of the application (or any of the reports along the way), they resulted in rushed development towards the end of the project. We should not be too hard in ourselves about this, however, as it is a frequent occurrence in industry for projects to be delayed or overrun their schedule (23). The critical underlying lesson is to be realistic about schedules and deadlines.

The team also learnt that it was beneficial to analyse the skills and abilities of the team members and to be realistic about each team member’s skills and abilities. The project management used the information to generate work schedules and assign tasks/roles to the members. It is important to be honest during the skills analysis process, otherwise roles are incorrectly assigned resulting in problems downstream.

Whilst the course has been useful in providing team members with an understanding of the Software Engineering process, at times the process has seemed sluggish as education was ongoing concurrently with the project, and at times the project has been held back because topics had not been covered in class. Thus it would have been useful for the education to have been completed in semester one, and the project started mid-way or towards the end of semester one, leaving the majority of semester two for project development.
References

(3) IEEE Standards (various, including):
   - IEEE Recommended Practice for Software Requirements Specifications (830-1998)
(4) W3C Specifications, http://www.w3.org/MarkUp/#recommendations
(10) Internet Dive Log, http://www.immersions.com/idl/
(17) WAI, http://www.w3.org/WAI/
(23) http://sunset.usc.edu/research/COCOMOII/TableofContentsPage2.htm
User Manual

Registration

1. When you first run the EchoBeach software you will be asked if you want to register for an account.

![Account Registration dialog]

Click 'Yes' to continue.

Note: If you have already registered you need to log in by clicking Edit->Preferences. Then click on the account tab. Check if your username and password are entered, if they are not, enter them and click ok.

2. You will then be asked to enter your personal details.

![EchoBeach - Configuration Setup Wizard]

Once you have entered you details, click 'next'.

3. You will then be asked to enter your Dive Details.
Once you have entered these details, click 'next'.

4. You will then be asked to enter contact details.
Once you have entered your contact details, click 'next'.

5. You will then be asked to enter your User Details.

Once you have entered your user details, click 'Finish' and you're done.
**Novice User**

The Dive Wizard is to be used to run through everything that is required before your dive information is uploaded to the web.

---

The navigation for the wizard is on the bottom of the screen and includes the buttons:

- **Cancel** – If you want to stop the process at any time you can click here.
- **Previous** – If you want to return to a previous screen to change some details this will allow you to do so.
- **Next** – Once you fill out any relevant details on a screen, the Next button will take you to the next screen.
- **Finish** – Once you have filled in all the details that you require, click on the Finish button to stop the wizard, then publish and/or save your Dive details.
Dive Data screen – search for your .csv file (the file from the dive computer) then Click Next to proceed.

On the following two screens you will be asked for information about the dive.
You will then be asked to select your dive buddies.

You can then enter a diary entry about the dive.

You can then choose pictures you wish to add to your blog.
Finally all that is left is to Publish your blog to the web or Save to Disk for later editing/publishing.

Importing your dive
1. Once you have registered for a new account, the Dive Blog Creator will appear.

2. To import a dive which you have uploaded from your dive computer either click the import button or go to the File menu and click the Import option (shown below).

3. Once you have done this a window like the one below will appear. Here you should locate the .csv file which holds all the data from your dive. Once you have found it click the Open button.
4. This will import all of the dive data from your dive and display it on the Dive Blog Creator screen. Now you will need to fill out some of the basic details about your dive. Firstly you need to add a new location, with the GPS coordinates included. To do this click on the Add Location button (shown below).

![Add Location button]

5. This will bring up a window like the one below. Into this enter the name of the location, the GPS coordinates and the other details which it asks for.

![Add Location window]

6. Once you have entered the details, click the Save button and the location will now be selected.

7. Within the dive rating section you have the chance to rate your dive. 1 is a poor dive, and 5 is an excellent dive. Use the drop down arrow to make your choice.

8. The rest of the information required on this page should already be filled in (if not feel free to enter it manually) so now go to the Buddies tab at the top of the page. This will bring up a screen like this:

![Dive Blog Creator window]
9. In the Registered Users box, all of the names of users registered with Echo Beach will appear. All you need to do is click on the name of the person, then click the < Add button. There name will then move into the Buddies box. To remove a buddy you just need to select them then click Remove >.

10. Now select the Dive Diary tab. A screenshot is shown below.

11. This is an area provided for you to write what you want relating to the dive. All you need to do is type in what you think and this will be published to the web page.

12. Next click on the Media tab. This page allows you to add pictures and videos which you took on your dive. To add a piece of media click the add button, here a window will appear asking you to locate the file you want to add to the website. Locate your picture then click the Open button. After you have done this the screen should look something like this:

13. The left hand box tells you the destination of the picture, clicking on the destination will let you preview the image. You can delete a picture by selecting it and then clicking Delete.
14. Now your dive is ready to publish on the Internet. To do this click the Publish button (shown below) or you can go to the File menu and click Publish.

15. Once you have done this a small window like the one below will appear. This is informing you of the upload progress.

16. Once you are informed of the completion of the upload, your Dive Blog will be up on the Internet. To view it go to the Echo Beach web page (http://www.csd.abdn.ac.uk/~teame/EchoBeach2/) and log in.

17. Congratulations! You have just created your first blog!

**Open a dive**

To open a dive you can either click the Open button on the toolbar or go to the File menu and click Open.

A window will then appear, asking you to locate the dive which you would like to open. Select the desired dive and click Open.

**Save a dive**

To save a dive in Echo Beach all you have to do is go to the File menu and click Save As .

A window will then appear, asking you what you would like to call your dive. Now all you have to do is type in the name and click Save.

Note: There is also a Save option on the toolbar and within the File menu. Use this to Save changes as you go, as it does not require a new file name every time. On the other hand, if you would like to save your dive with another name you will need to use the Save As option. Remember to keep saving your work as you go!
**Personal Details**

Your personal details will be saved to system database when you first register, if you want to change your details click on Personal Details from Edit menu bar, the following window will pop up with your registered details:

You simply change whatever you want to change and click on Apply, the information you changed will update automatically. Click on Cancel if you don’t want to apply your changes to the database.
Preferences

Preferences contains database, folder and account details. Click Preferences from Edit menu bar, the following window will pop up:

These are the details about the database your system is using, it was generated by the configuration file when you started to run the software. Username and Password here means your database access login name.

Click on the folder tab, it will show something similar to this:
Again, the configuration file generated this when you set up the system, the Home Folder is where you store the software on your disk drive, you can change that by clicking on Browse. A window similar to the one below will appear:

You can change the directory to access the required folder. Next select the folder and click Open, the Home Folder will be changed to the directory you selected.

Click on the Account tab, this shows your account Username and Password.
You can change your password by click on Change Password, a dialog will pop up ask you to type your current password and new password.

Be aware, a warning such as the one in red above, will appear if you do not satisfy the criteria with your password choice. All passwords must be between 4 and 12 characters. Click on Save to change your password. Click on Cancel if you want to keep your original password.
Maintenance Manual

Installation
To install the Client:
- Under windows run the installation program.
- Under Unix/Linux extract the tar.gz file to the desired installation directory. Move the echobeachconfig.xml to your home directory.

To install the Blogging Engine:
- Extract the tar.gz file to the Web Server page directory (e.g. into public_html)
- Edit the config/config.php file to suit your environment

Compilation
Client:
- Compile all the Java files contained in the package. It is preferred that Eclipse is used for project management/maintenance.

Blogging Engine:
- This is written in a scripting language, thus no compilation is required.

External Dependencies
Client:
- MySQL JDBC Connector (available from http://www.mysql.com). This should be installed into the same directory as the EchoBeach.jar file.

Blogging Engine:
- PHP must be compiled with GD and MySQL support.

Source Code - Client
com.echobeach.core
- ChangePassword.java – the change password dialog
- FeedbackWindow.java – the feedback dialog
- HelpWindow.java – the help window
- MainWindow.java – the main GUI
- PersonalDetails.java – the personal details dialog
- Preferences.java – the preferences dialog
- Runner.java – the application launcher
- Splash.java – the splash screen window
- SplashScreenCanvas.java – the splash screen picture canvas

com.echobeach.core.beans
- BlogBean.java – bean representing (& processing) a blog
- BuddiesBean.java – bean representing (& retrieving) a buddy
- DataPointBean.java – bean representing a data point
- LocationsBean.java – bean representing (& managing) a location
- PersonalDetailsBean.java – bean representing (& managing) personal details
- RegistrationBean.java – bean representing (&creating) a registration
com.echobeach.novicewizard
- AddLocation.java – the add location dialog
- NoviceScreen1.java – novice wizard panel 1 (CSV)
- NoviceScreen2.java – novice wizard panel 2 (Data input 1)
- NoviceScreen3.java – novice wizard panel 3 (Data input 2)
- NoviceScreen4.java – novice wizard panel 4 (Buddies)
- NoviceScreen5.java – novice wizard panel 5 (Diary)
- NoviceScreen6.java – novice wizard panel 6 (Media)
- NoviceScreen7.java – novice wizard panel 7 (Save/Publish)
- NoviceScreen8.java – core UI dive overview panel

com.echobeach.setupwizard
- SetupScreen1.java – setup wizard panel 1 (Personal Details)
- SetupScreen2.java – setup wizard panel 2 (Dive Statistics)
- SetupScreen3.java – setup wizard panel 3 (Contact Details)
- SetupScreen4.java – setup wizard panel 4 (Account Details)

com.echobeach.utils
- Configuration.java – configuration system (for echobeachconfig.xml)
- CustomFileFilter.java – file filter for dialog boxes
- DatabaseConnector.java – database access
- FileOpen.java – file reader
- FileSave.java – file writer
- FileUtilities.java – misc. file read/write facilities
- ImagePreview.java – image preview panel
- MissingSectionException.java – exception for Configuration
- MissingSettingException.java – exception for Configuration

com.echobeach.GraphDraw
- GraphDraw.java – generates a JPEG graph of dive depth/time

com.echobeach.wizardEngine
- EmptyPanel.java – empty panel, used internally to Wizard
- SpringUtilities.java – public domain UI class for Wizard buttons
- Wizard.java – the main UI for the wizard
- WizardScreen.java – interface describing wizard panels
Source Code – Blogging Engine

- access.php – controls login/logout functions
- changePass.php – change password
- comments.php – make a comment, read a dives comments
- createThumbnail.php – create a thumbnail from a BLOB stream
- datebox.inc – the drop-down date menu
- divedata.php – the blog dive data page
- divediary.php – the blog dive diary page
- diveoverview.php – the dive overview for the blog
- diverProfile.php – the profile of a diver
- downloads.php – the downloads available from this site
- error404.php – a customised 404 page
- faqs.php – Site FAQ’s
- forgotpass.php – issue a new password
- forgotpwd.php – issue a new password
- functions.php – various data retrieval & processing functions
- home.php – the welcome page
- index.php – the main page, wraps around all content
- logoff.php – log off of the site, destroys cookies
- manageProfile.php – manage the users profile
- media.php – the media elements for a blog
- memberhome.php – the members home page
- myComments.php – shows diver their recent comments
- myDives.php – divers recent dives
- registrationform.inc – the registration form
- searchblogs.php – search all blogs
- showPhoto.php – show a photo
- showVideo.php – show a video
- signup.php – register for an account
- site.css – the CSS styling for the site
- termsandconditions.html – the site terms and conditions
- viewblogs.php – views a blog, manages the correct tab highlighting & page includes
- viewphoto.php – view a photo
- config/config.php – all the site configuration settings
- database/mysql.php – MySQL database access
- i18n/EN.php – the start of internationalisation support