Summer 2010

Implementing Functionality to Identify and Fix Time Gaps, and Generate Custom Reports and Graphs Will Improve the Analysis, Modification and Reporting Off Tidal Data in Ireland

Shane Galvin
Regis University

Acknowledgement

Thanks to administration and faculty staff of Regis University and the National University of Ireland, Galway for the opportunity to further my education A special thanks to Doug Hart for his advice and guidance throughout the year.

To Guy Westbrook for the opportunity to work on this project and for his support and feedback for the duration of it.

And finally would like to thank my family and friends for their support and patience over the last two years.

Table of Contents

List of Abbreviations

ASP	Active Server Pages
BIZ/BLL	Business Logic layer
CSS	Cascading Style Sheets
DAL	Data Access Layer
FDD	Feature Driven Development
GIF	Graphics Interchange Format
HTML	HyperText Markup Language
IIS	Internet Information Services
ITGN	Irish Tide Gauge Network
LAN	Local Area Network
LINQ	Language Integrated Query
MI	Marine Institute
ODBC	Open Database Connectivity
OSS	Ocean Science Services
PDA	Personal digital assistant
PNG	Portable Network Graphics
RDBMS	Relational database management system
RTDI	Marine Research, Technology Development and Innovation
SQL	Structured Query Language
XP	Extreme programming

List of Tables

List of Figures

Abstract

The Irish National Tide Gauge Network (ITGN) is on-going development involving the Marine Institute (MI) and a number of public sector organisations to develop a permanent tidal monitoring infrastructure. As part of this on-going work the MI wish to be able to analyse the existing ITGN dataset for time gaps, and make changes to the dataset if needed, they would also like to be able to view custom reports and graphs based on the ITGN dataset. This thesis presents a web-based and windows application to meet the aforementioned needs of the Marine Institute and ITGN.

Chapter 1 – Introduction

The Marine Institute (MI) is the national agency responsible for Marine Research, Technology Development and Innovation (RTDI) in Ireland. The Institute was set up under the 1991 Marine Institute Act with the following role: *"to undertake, to co-ordinate, to promote and to assist in marine research and development and to provide such services related to research and development that, in the opinion of the Institute, will promote economic development and create employment and protect the marine environment"* as stated by the Marine Institute (2009). The Marine Institute's headquarters are located at Oranmore, Galway, and it also has regional offices and laboratories in Dublin and Mayo.

The Irish National Tide Gauge Network (ITGN) is an on-going development involving the Marine Institute and a number of public sector organisations to develop a permanent tidal monitoring infrastructure, that will ultimately consist of between 35 and 40 tide gauge stations. As of July 2009 the ITGN has 16 operational tide gauge stations located along the coastline of Ireland, each containing a series of instruments measuring the likes of water level, temperature, and atmospheric pressure. Periodic readings are taken from these instruments on the tide gauges and recorded in a database maintained by the Marine Institute.

Statement of the Problem and Goals to be achieved

As part of the on-going enhancements to the ITGN, the MI wish to develop both a web-based application to improve the reporting and modification of Irish tide gauge data, and also a windows based analysis application to help identify time gaps in the tidal dataset. This new application will be used by the Ocean Science Services (OSS) dept of the MI and will focus on the following three key functions in relation to the Irish tide gauge data:

- Analysis

- Modification
- Reporting

Windows Application Features:

Analysis

The OSS department would like to know about time gaps in the data within a short time period of it being collected. Time gaps can be caused by the likes of instrument malfunctioning on the tide gauges, or communication problems in transmission between the tide gauge and the base station. An analysis engine will be developed as a Windows Service that will be run periodically and detect any gaps in the data and send an email report to authorised users of the OSS team. Currently the only method of detecting time gaps in the data is by manually scrolling through the tidal observations that are displayed to the MI's public website, or sometimes visitors to the site would report on any time gaps they encountered. Manually checking for these time gaps can be tedious, error prone and time consuming.

Web Application Features:

The new web application must provide the following features in both and intuitive and user-friendly manner.

Modification

Currently the only method of updating the tide gauge readings is at the database table level, so only members of the applications development team within the MI are authorised to make amendments to the data. Authorised users from the OSS department will need to be able to make amendments to the dataset to facilitate missed readings or data loss when possible. The application will need to allow admin users to amend the dataset via web forms, and also validate the data before being persisted to the database.

Reporting

The reporting feature will be separated into two key sections, data reports and tidal graphs.

Data Reports

The web application must provide a user-friendly interface to query the tidal data based on factors such as date range and tide gauge location and have these reports exportable to both Word & Excel formats. These reports should only be viewable by authorised users who have successfully logged into the web application.

Tidal Graphs

The MI currently uses two different graphing components, the first using crystal reports and another using a flash based solution. Both of these require licenses and some users using older model of PDA's and cellular phones have been unable to view the flash based graphs as flash plugins were not supported on these devices. The MI would like to replace the existing graphing components with a new component that does not require a license fee and which can render a graph as an image such as JPEG or a GIF to make it more portable with older mobile devices. This graphing component must also utilize the same user-friendly interface as the data reports allowing users to render a graph based on a selected date range and tide gauge location. These graphs should also be accessible from outside of the web application by users who have knowledge of the graph generator URL and the parameters required.

Challenges and Barriers to Success

For the project to be a success the following criteria must be achieved:

- The analysis engine can detect time gaps during a set date range.
- Admin users can easily make amendments to the existing Irish tides dataset using web forms.
- Customised reports can be generated on the data and exported to a variety of file formats from within the web application.
- Tidal graphs can be rendered as an image without the need for crystal reports libraries or flash plugins, and also be accessed via the URL from outside the web application.

The delivery of the application also has a hard deadline of 17th September 2010. Activities will need to be delivered on time at all stages and the project will need to be tightly controlled and managed. Decisions will need to be made early on in the project lifecycle as which graphing component will be used. Other 3rd partly controls and libraries will also need to be investigated in advance.

Organization of this paper

A brief summary of each of the chapters in this thesis:

Chapter 1 – Gives a brief introduction to the problem being addressed in this paper, and how the goals can be achieved, and the challenges and barriers that will have to be faced.

Chapter 2 – Describes the agile SCRUM methodology used to design and develop the project, and how the tasks were organized and scheduled.

Chapter 3 – Describes the N-tier architecture that was used for the design of this project and also the technologies used.

Chapter 4 – Introduces some of the 3rd party controls that were used as part of the development and also how I selected the graphing component that will be used for generating the tidal graphs.

Chapter 5 – This chapter provides and overview of the Irish Tides web application and Tidal Analyser Service and how each of the user stories described in the *Irish Tides User Stories* section have been solved in the application. It also shows some of the GUI's used in the web front end.

Chapter 6 – Evaluates my thesis statement, and determined if it meets the goals I set out with in the beginning.

Chapter 7 – Describes the project history, the origins of the project and schedule of events.

Chapter 8 – Is the concluding chapter where I discuss the lessons that I have learned, what I would have done differently, and what's next for the project.

Chapter 2 – Project Methodology

In this chapter I will discuss the software project methodology used for the design and development of the Irish tides application. I used agile methods and in particular the SCRUM process in which to manage and control the development of the Irish tides application using iterative, and incremental practices.

What is Agile?

The Agile Methodology (2008) describes agile as *"an approach to project management, typically used in software development. It helps teams respond to the unpredictability of building software through incremental, iterative work cadences, known as sprints"*. The name agile was coined by a group of software professionals, all of whom had an interest in lightweight software methodologies, and came together to form the organisation the Agile Manifesto, a group that promotes agile software development practices.

The Agile Manifesto (2001) values the following:

"Individuals and interactions *over processes and tools*

Working software *over comprehensive documentation*

Customer collaboration *over contract negotiation*

Responding to change *over following a plan" [3]*

The Agile Manifesto believes that there is value in the items on the right that are listed above; it's just that they value the items on the left more.

Individuals and interactions *over processes and tools*

The manifesto places a higher value on the individuals and interactions within the project team, than on the processes and tools they will use during the project. For the project to be a success the team must carefully select the tools and processes that best suit the goals of the project and the technologies being used.

Working software *over comprehensive documentation*

The manifesto recognize that no amount of documentation no matter how comprehensive has any value if the software does not do what it is supposed too. The priority should be on writing code and having a working system than on documenting all of your changes along the way.

Customer collaboration *over contract negotiation*

Agile development teams focus on the needs of the customer by working closely with them for the duration of the project.

Responding to change *over following a plan*

Change is inevitable in software development, agile development emphasis practices that facilitate teams to adjust and react to changing requirements.

The Agile Manifesto (2001) also follows the following 12 principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

According to Softhouse Consulting (2009) *"A central concept for agile methods is adaptation to changing external factors. Where older methods are predictive and attempt to foresee future needs, the agile methods are adaptive and quickly adapt to new demands, adhering to the "Embrace change!" motto. The only measurement of success is functioning products".*

Agile development is an umbrella term used to describe a group of methodologies of which some of the more prominent ones are Agile Modeling, Crystal, Dynamic Systems Development Method (DSDM), Extreme programming (XP), Feature Driven Development (FDD), Scrum, and Lean software development. For this project I am just going to concentrate on the one of these methodologies, and that is SCRUM.

SCRUM

SCRUM was first developed in the early 1990's and gets it's name from the rugby activity, which involves the team getting together shoulder-to-shoulder to try to move the ball forward. As an agile process it is focused on delivering business value to customers by allowing teams to choose the amount of work to be done and decide how best to get it done. In scrum you prioritise work based on business value, and deliver working software in short iterations.

According to Schwaber & Sutherland (2007) *"Scrum is designed to add energy, focus, clarity, and transparency to project planning and implementation"* and it will also

- Increase speed of development
- Align individual and corporate objectives
- Create a culture driven by performance
- Support shareholder value creation
- Achieve stable and consistent communication of performance at all levels
- Enhance individual development and quality of life

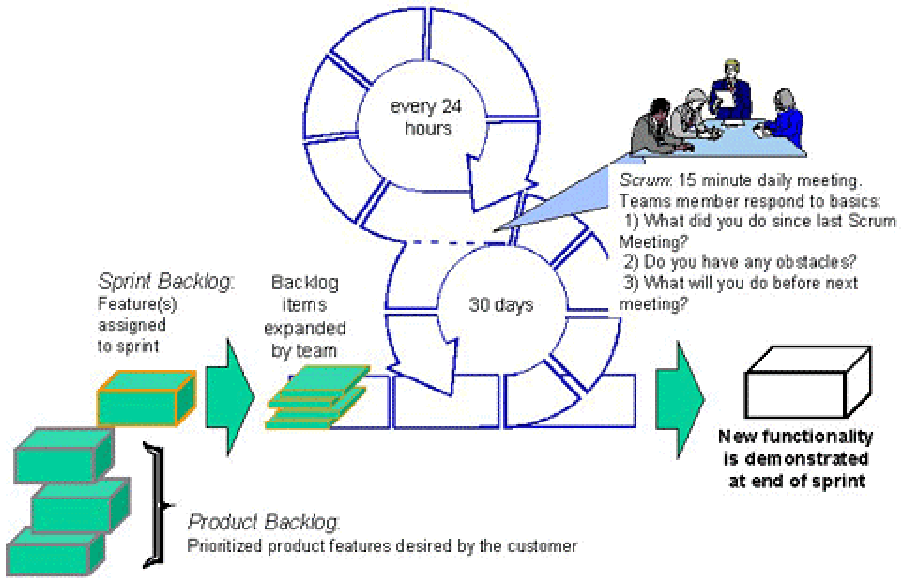

every 24 hours
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
30 days
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Figure 1 Scrum Process Flow, Control Chaos (2007)

Prepare for SCRUM

The following are some of the key components of a SCRUM project as shown in *Figure 1 Scrum Process Flow, .*

Product/Project Backlog

The project backlog is where I place all of the user stories I have gathered from meetings with Guy Westbrook of the OSS dept within the MI. It is essentially a list of features or requirements that the customer would like, described in the customers own words.

Sprint Backlog

A sprint backlog is where the highest priority user stories from the project backlog are placed that will be included in the latest sprint.

Sprint

A sprint or iteration consists of the sprint backlog items that will be undertaken next, and can be completed within the duration of the sprint, which will be the current calendar month the sprint is to be undertaken during. At the end of each sprint I will review all of the items that were completed in the sprint backlog, and identify and prioritise tasks for the next sprint.

SCRUM Meeting

As I am undertaking this project on a part-time basis and outside of my work hours it is not feasible to have a SCRUM meeting every day. I therefore set aside two meetings a week in which I can asses what I have done since the last meeting, what I will do next, and is there anything that may prevent me from doing what I had planned.

Irish Tides User Stories

Detecting time gaps

The applications should be capable of detecting time gaps for each of the instruments on each tide gauge for a 24-hour period previous to the latest reading for that tide gauge. OSS users should receive an email report every day detailing any time gaps that were detected.

Web Logon

OSS users must enter a username and password to gain access to the Irish tides web application.

Graph Reporting

Once logged in through the web application users should be able to generate a tidal graph by selecting a tide gauge station, instrument and a date range. The graph should open in a new window and should be rendered as an image. Users should also be able to select a width, height and choose from one of the following 3 styles for the graph, White background with Blue curve, Blue background with White curve, and Blue background and Yellow curve.

Data Reporting

Once logged in through the web application users should be able to generate a tidal data report in either Microsoft Word or Excel formats, by selecting a tide gauge station, instrument and a date range.

Observations Edit/Delete

Once logged in through the web application users should be able to amend/delete any of the observations. Users must first select a tide gauge station, instrument and a date range and select search. They should then be presented with a list of observations in a list view, and should then have the option to edit or delete any of the observations.

Observations Add New

Once logged in through the web application users should be able to add a new observation by selecting a tide gauge station and instrument, and then entering a date, reading and any comments (optional).

Estimates

Each of the user stories outlined previously was then broken up into tasks and a time estimate is assigned to the user story.

Table 1 – Estimated Project Schedule

User Story	Tasks	Estimate
Project Research	• Research suitable architecture framework to be used. • Research ASP.NET and C#, and SQL Server 2005 technologies.	2 weeks
Research & Select Graphing Component	• Research graphing components available for ASP.NET. • Test 3 or 4 graphing libraries for suitability to the MI's tidal graphing requirements. • Select a graphing component to be used.	3 weeks
Set-up Environment	• Set-up database using scripts obtained from MI. • Create development environment in Visual Studio.	1 week
Create DAL	• Research LINQ to Entities. • Create a Data Access Layer based on the Irish Tides Database. • Create Stored Procedures.	3 weeks
Create BIZ	• Create a BIZ layer for which each object can select, save or delete. • Implement list functionality for each BIZ object.	3 weeks

Detecting time gaps	• Develop windows service. • Implement functionality to detect time gaps. • Add functionality to send email reports containing the time gaps to the OSS users. • Update BIZ & DAL layers.	4 weeks
Web Logon	• Create web form for user logon, log-off pages. • Create the default page that the user will be brought once they have been validated. • Create a master page to be used on all pages once the user has been validated.	2 weeks
Data Reporting	• Develop a web form to allow users to specify criteria for the report they wish to generate. • Develop a web form to generate Microsoft Word reports. • Develop a web form to generate Microsoft Excel reports.	3 weeks
Graph Reporting	• Amend the data reports web form to allow for graph reports. • Develop a web form to generate tidal graphs as images.	4 weeks
Observations Edit/Delete	• Develop a web form to enable users to edit or delete observations. • Create/Update Stored Procedures for edit and delete functionality.	2 weeks
Observations Add New	• Develop a web form to enable users to add new observations. • Create/Update Stored Procedures for adding new observations.	1 week
Thesis	• Write up Thesis	4 weeks

Sprints

I then prioritised the estimates and added them to sprint backlogs. As a sprint duration was a calendar month Sprint 1 would start in January 2010, and have an expected completion date of the project for the end of August 2010. Testing is considered a vital part of each sprint and time allocated for testing has been factored into all estimates.

Sprint 1:

- Project Research
- Research Graphing Component

Sprint 2:

- Select Graphing Component
- Set-up Environment
- Create Data Access layer

Sprint 3:

- Create Data Access layer
- Create Business Access Layer

Sprint 4:

- Create Logon
- Create Tidal Analyser Windows Service

Sprint 5:

- Tidal Analyser – Email Component
- Data Reporting

Sprint 6:

- Graph Reporting

Sprint 7:

- Observations Add/Edit/Delete

Sprint 8:

- Write up Thesis

Chapter 3 - Architecture

The underlying architecture for this project is based on an N-tier layered model, which is the most common architecture model for designing small to medium sized enterprise web applications.

Platform Architecture

The MI has a requirement that the application be developed using the Microsoft .NET Framework 3.5, and the back-end database to be SQL Server 2005. This was to coincide with in-house developer experience, and also so that it can be deployed on a Microsoft Windows Operating System.

.NET Framework 3.5

According to Microsoft (2009) *"The .NET Framework is Microsoft's platform for building applications that have visually stunning user experiences, seamless and secure communication, and the ability to model a range of business processes. The .Net Framework consists of:*

- *Common Language Runtime – provides an abstraction layer over the operating system.*
- *Base Class Libraries – pre-built code for common low-level programming tasks.*
- *Development frameworks and technologies – reusable, customizable solutions for larger programming tasks" [18].*

The .NET Framework 3.5 is the latest in the series of frameworks that has been released by Microsoft. Although each of the frameworks support multiple programming languages via the CLI (Common Language Infrastructure), in this project we will just concentrate on the most popular of these languages, C#.

Goals of building an N-tier application in .NET

Sheriff (2002) lists the following goals that an n-tier application design should seek to achieve.

- If you change the underlying data access methods, the client-side code should not have to change.
- All data access routines should be exposed as objects instead of function calls. As an example, it is much easier to use ADO than the ODBC API calls.
- SQL should be eliminated from the client-side code. The client code should just be concerned with methods and properties.
- Table and column names should be eliminated from the client-side code. Typed datasets can present table and column names as properties, providing an IntelliSense list, as opposed to having to type in a string name. This means at compile time, checks can be made for data types and names of columns.
- The client code should not care where the data comes from. It should just care that it can retrieve and modify the data in some object and the object will take care of the details.
- The coding you need to do on the client side should be simplified. Instead of using many functions, your application should be able to use objects with properties and methods.
- It becomes easier to create and use the classes than the function calls.
- It becomes easier to add functionality to your applications, and change the functionality, without breaking the client-side code.

Visual Studio 2008

Visual Studio 2008 is the latest IDE (Integrated Development Environment) from Microsoft and can be used to develop a variety of different types of applications for the .NET Framework. It contains a built in code editor, debugger and GUI designer. *Figure 2 Irish Tides Visual Studio Solution Explorer* shows the Irish Tides application in the solution explorer window within visual studio. The Irish Tides solution contains the following three project types:

1. **IrishTidesDomainModel** – A visual studio class library project that contains the Business Logic Layer (BIZ) code in the namespace IrishTides.Biz, and the Data Access Layer (DAL) code in the namespace IrishTides.Dal.
2. **IrishTidesWebApp** – A visual studio ASP.NET web application project that contains the front-end web forms, Style Sheets and images. The IrishTidesWebApp also references the IrishTidesDomainModel class library, the AjaxControlToolkit library and a 3rd party ZedGraph graphing library.
3. **TidalAnalyser** – A visual studio windows service application for detecting time gaps in the data and emailing reports on this to selected users. The TidalAnalyser also references the IrishTidesDomainModel library and the 3rd party log4net logging library.

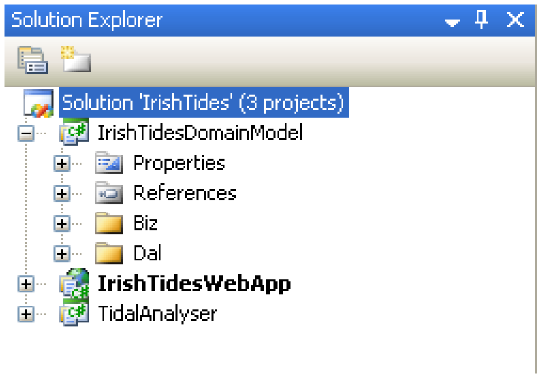


Figure 2 Irish Tides Visual Studio Solution Explorer

SQL Server 2005

SQL Server 2005 is a client-server relational database that according to Machanic (2006) *" is an enterprise-ready database platform that provides high performance, scalability and availability, plus a vast array of easy-to-use features, all at a low price point"*.

N-tier Architecture

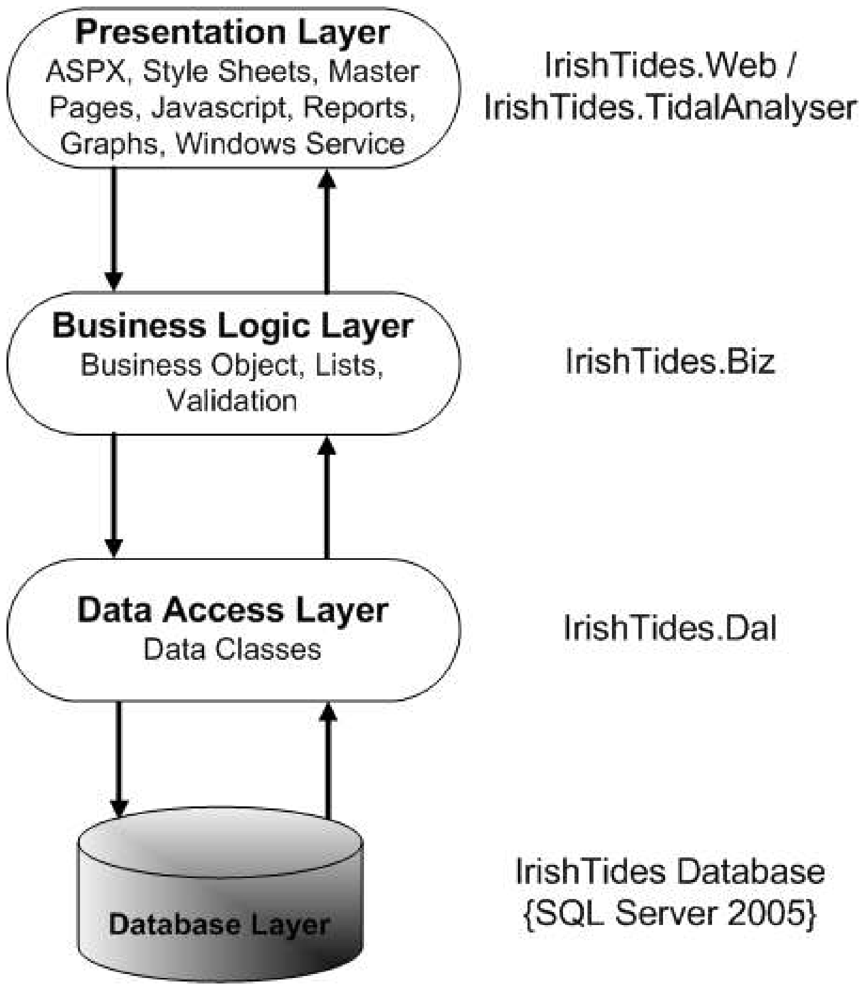

Presentation Layer
ASPX, Style Sheets, Master Pages, Javascript, Reports, Graphs, Windows Service
IrishTides.Web / IrishTides.TidalAnalyser
Business Logic Layer
Business Object, Lists, Validation
IrishTides.Biz
Data Access Layer
Data Classes
IrishTides.Dal
Database Layer
IrishTides Database {SQL Server 2005}

Figure 3 N-tier layer architecture for Irish Tides

The N-tier architecture is a client-server architecture, and for this project it will be separated into four distinct tiers or layers, which are the presentation layer, the business layer, the data access layer, and the database layer. As shown in *Figure 3 N-tier layer architecture* *for Irish Tides* each layer interfaces with the layer directly below it, and each layer has it's own functions and responsibilities to perform.

Advantages of using an N-tier architecture

The following are some of the advantages of a well-designed application using N-tier architecture:

- Clearly defined and separated layers with each layer requiring specific knowledge i.e. the presentation layer's only concern is the display of the data and the developer need not have any knowledge of the underlying database.
- You can change database servers without having to make changes to any of the layers above the data access layer.
- You can change the presentation layer from a web application written in ASP.NET to a windows forms application without having to make changes to the business logic or data access layers.

Presentation Layer

The presentation layer is the uppermost level of the application and is the layer that the user will interact with and that will display data to the user, which is usually through a web browser or a windows form. In this project the presentation layer is separated into two components, a front-end website and a windows service. The presentation layer works with data objects retrieved from the business layer to transform this data into something that can be read and understood by the end user, which in the case of this web site is web pages that can be viewed though a browser, and for the windows service is emails that can be opened in an email client.

Front-end Website

In this project the front-end website consists of ASP.NET web forms (.aspx), HTML pages, JavaScript, CSS style sheets, Master Pages, and also web user controls (.ascx), and also the rendering of Word and Excel Reports, and Tidal Graphs using the presentation layer. The web browser or web client communicates with ASP.NET through the Microsoft Internet Information Services (IIS) web server as shown in *Figure 4 Communications with Presentation Layer*.

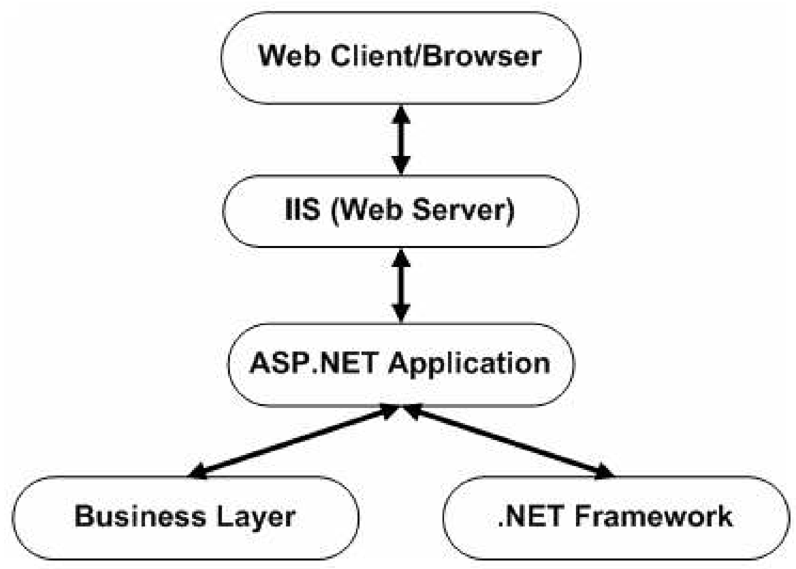


Figure 4 Communications with Presentation Layer

The Presentation layer for Irish Tides will consist of standard ASP.NET web controls such as Labels, TextBox's, DropDownList's and Buttons, with the addition of some rich UI Controls provided by the AjaxControlToolkit for date display and validation such as the CalendarExtender, MaskedEditValidator and MaskedEditExtender.

To display the Irish Tides reports as Word or Excel Documents we set the appropriate MIME content type and leave it up to the browser to display it to the end user in this format. For the reports to appear in the correct format the end user or client must have Microsoft

Word and Excel installed or at the very least Word or Excel viewer applications. For example to display a Word document we set the content type of the response object as follows

```
Response.ContentType = "application/msword";
```

And to display an Excel document we set the content type to

```
Response.ContentType = "Application/x-msexcel";
```

To display the tidal data in report form in a word document we bind the results set to a DataList control and apply the appropriate styles to the DataList in the ASPX page responsible for generating the word report. For excel we need to do things slightly different and build up a comma separated string list of items and then write the contents of this string to the response stream.

Tidal Analysis Service

The Irish Tides tidal analysis service is developed as a Windows Service. According to the Microsoft MSDN (2009), windows services *"enable you to create long-running executable applications that run in their own Windows sessions. These services can be automatically started when the computer boots, can be paused and restarted, and do not show any user interface"*. Windows services provide no user interfaces and instead run in the background and be set to start when the computer boots and do not require a user to be logged on.

The Irish Tides TidalAnalyserService must implement the System.ServiceProcess.ServiceBase class and override the Dispose, OnStart and OnStop methods.

```
public class TidalAnalyserService : ServiceBase
{
    protected override void Dispose(bool disposing)
            {
            // dispose of any managed resources
    }
```

```
        protected override void OnStart(string[] args)
        {
            // code for starting the servie
        }

        protected override void OnStop()
            {
                // code for stopping the servie
        }
    }
```

Business Logic Layer

The business logic layer contains the business logic or business rules for the application. As each layer should only interface with the layer directly below it in an N-tier architecture, the presentation layer can therefore not communicate directly with the data access layer, so the business logic layer then acts as a mediator for the transfer of data between the presentation and data access layers. In the business logic layer you should define a class to represent each database table (as shown in *Figure 5 N-tier - Business Logic Layer*), and a property for each column in the table, and also any validation rules that need to be applied. The business logic layer should have no concern about how to present data on the web or similarly should have no knowledge of the database being used or how to access it. This clear separation of concern means that you would be able to change from an ASP.NET web application to a windows forms application without having to change the business or data access layers, or you could also change to a different database server by only having to update the data access layer. In this application the presentation layer consists of both a web front-end application and a windows service, which is only made possible by having a separate business logic layer which can then is reusable between both presentation layer components.

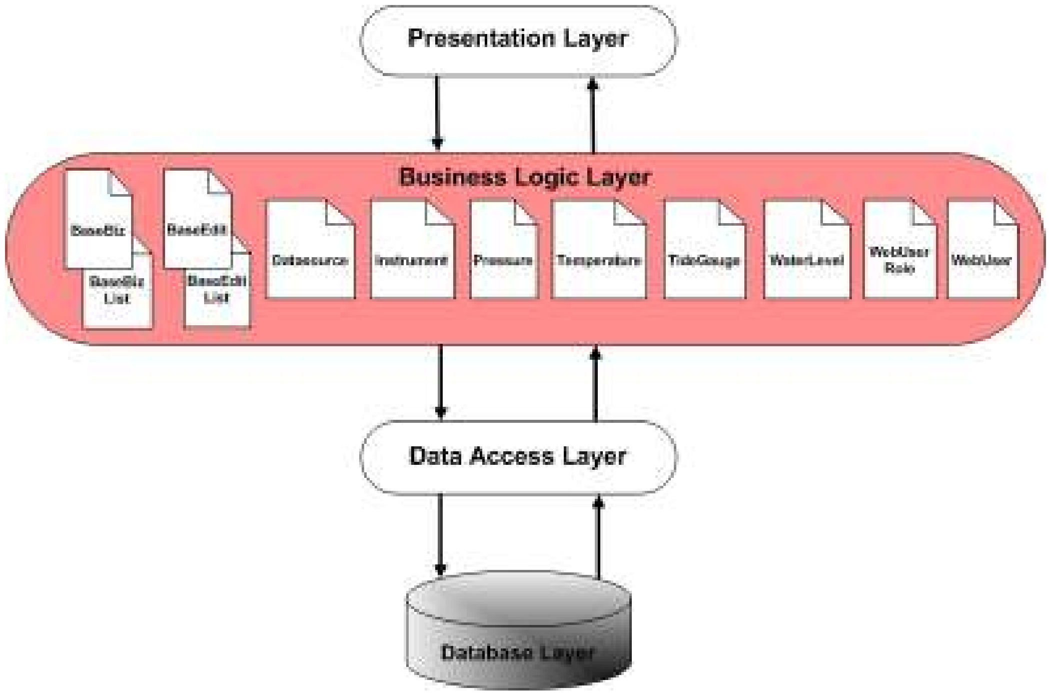


Figure 5 N-tier - Business Logic Layer

Data Access Layer

The data access layer according Varallo (2009) *"manages communication between the database and the business logic layer … The goal of the DAL is to create classes that expose methods that enable the business layer to retrieve or persist data to the database"*. The data access layer contains all of the components you need to access the database such as tables, views, stored procedures, indexes, and helps provide a level of abstraction for the database structure. For this project I will be using LINQ to Entities to map the database tables to C# classes as shown in *Figure 6 N-tier - Data Access Layer*. LINQ to SQL is an ORM (object relational mapper) tool that comes as part of the .NET framework, and allows you to model a relational database using .NET classes by using a DataContext object as the central point to query data from the database. LINQ expressions can then be used to query the database, as well as to update/insert/delete data.

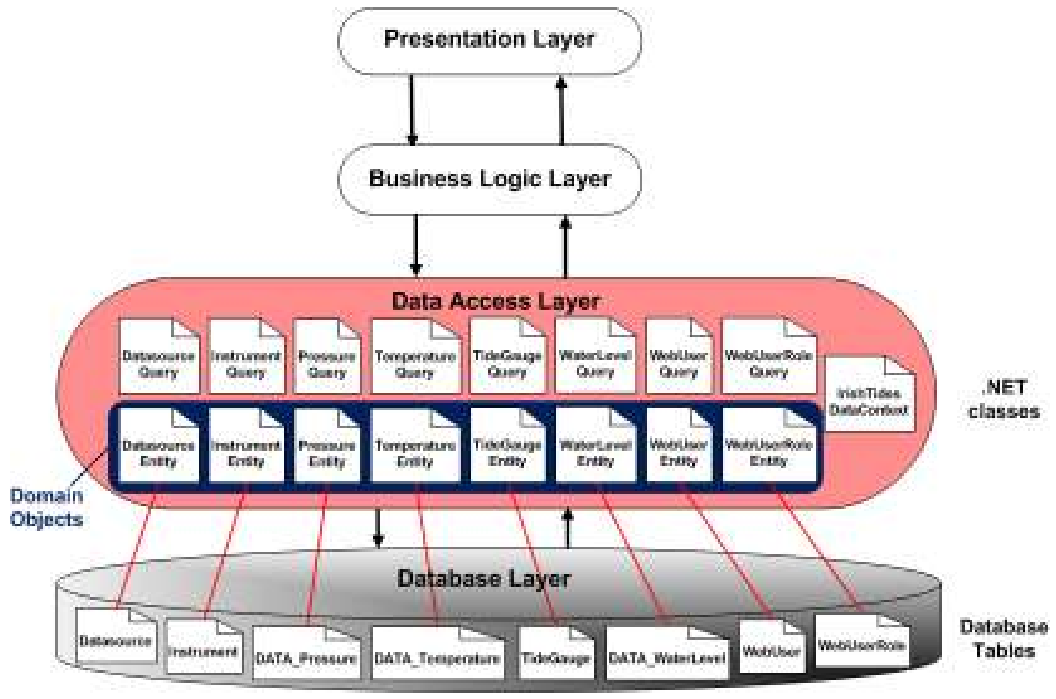


Figure 6 N-tier - Data Access Layer

LINQ

According to the MSDN: Linq (2009)*"Microsoft Language Integrated Query (LINQ) offers developers a new way to query data using strongly-typed queries and strongly-typed results, common across a number of disparate data types including relational databases, .NET objects, and XML."*. LINQ is a rich SQL like syntax that can query any object that implements the IEnumerable<T> interface.

Ferracchiati (2008) points out that the three basic steps required for LINQ to SQL to be able to interact with a SQL Server database are:

1. Create classes for the tables in the database that you want to use, decorating them with appropriate LINQ attributes. These classes are usually called entities.
2. Decorate the fields and properties in these classes so LINQ can use them and knows how to use them.

3. Create a DataContext object to mediate between the database tables and the classes that map to them.

LINQ to SQL Entities

The MSDN: LINQ to Entities (2009) *"The Entity Data Model (EDM) is a conceptual data model that can be used to model the data of a particular domain so that applications can interact with data as entities or objects"*. LINQ to Entities is part of the ADO.NET Entity Framework, and it enables you to write database queries in languages such as C#, the benefit of this is that you can then concentrate on working with domain specific objects and not have to worry about the underlying database structure. An entity class (domain object) is a class that is mapped to an SQL Server database table using LINQ to SQL and the entity object can then be passed back up to the business logic layer. As highlighted in *Figure 6 N-tier - Data Access Layer* each database table is mapped to an entity class, each entity class name is postfix with the word entity, and each entity also has a query class postfix with the word query which contains some methods to perform for that entity such as select and delete operations.

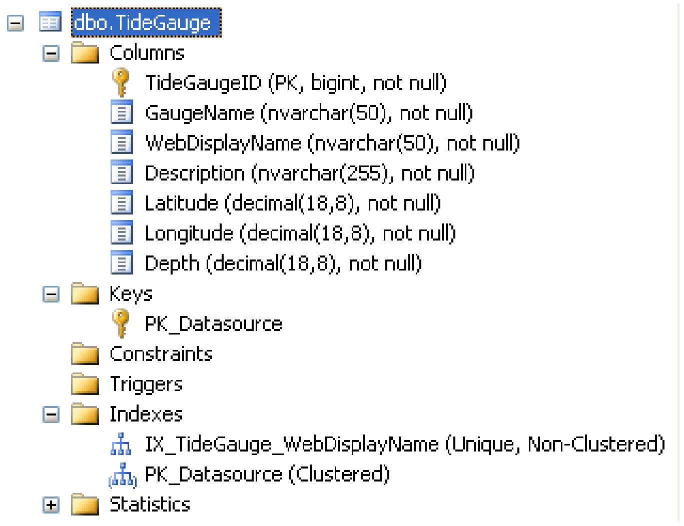

```
dbo.TideGauge
  Columns
    TideGaugeID (PK, bigint, not null)
    GaugeName (nvarchar(50), not null)
    WebDisplayName (nvarchar(50), not null)
    Description (nvarchar(255), not null)
    Latitude (decimal(18,8), not null)
    Longitude (decimal(18,8), not null)
    Depth (decimal(18,8), not null)
  Keys
    PK_Datasource
  Constraints
  Triggers
  Indexes
    IX_TideGauge_WebDisplayName (Unique, Non-Clustered)
    PK_Datasource (Clustered)
  Statistics
```

Figure 7 TideGauge Database Table

Figure 7 TideGauge Database Table shows the TideGauge database table viewed from Microsoft SQL Server Management Studio. The full table name is “dbo.TideGuage” and contains the following columns TideGaugeID (PK), GauageName, WebDisplayName, Description, Latitude, Longitude, and Depth.

The following code snippet shows how an entity class can be mapped to the database table dbo.TideGauge to create the TideGauge domain object, and how the column name TideGaugeID within that table is mapped to the ID property in C#.

```
[Table(Name = "dbo.TideGauge")]
public class TideGaugeEntity
{
    private long _TideGaugeID;

    [Column(Name = "TideGaugeID", IsPrimaryKey = true,
        DbType = "BigInt NOT NULL IDENTITY")]
    public string ID
        {
        get
        {
            return this._TideGaugeID;
        }
        set
        {
            if ((this._TideGaugeID != value))
            {
                this._TideGaugeID = value;
            }
        }
    }
```

The Table and Column objects are new to .NET 3.5 and are included in the System.Data.Linq.Mapping namespace.

As mentioned earlier each entity class also has a query class, which allows you to query your domain objects. In this class for instance we only want users to be able to select data from the database so we expose a single Select method that takes an id as a parameter and passes this id on to the GetTideGaugeById method within the DataContext class, more on this in the *Data Context* section.

```
public class TideGaugeQuery : BaseDal<TideGaugeEntity>
{
    private string _connString = null;
    public TideGaugeQuery(string connString)
    {
        _connString = connString;
    }

    public override List<TideGaugeEntity> Select()
    {
        using (IrishTidesDataContext db = new
             IrishTidesDataContext(_connString))
        {
            return db.GetTideGauges().ToList();
        }
    }

    public override TideGaugeEntity Select(long id)
    {
        using (IrishTidesDataContext db = new
             IrishTidesDataContext(_connString))
        {
            return Select(db, id);
        }
    }

    public TideGaugeEntity Select(IrishTidesDataContext db,
         long id)
    {
        ISingleResult<TideGaugeEntity> result =
             db.GetTideGaugeById(id);
        return result.SingleOrDefault();
    }

    public override void Delete(long id)
    {
        throw new NotImplementedException("Delete
             not implemented");
    }
}
```

Data Context

A DataContext is used to connect to the database, to execute query commands to retrieve data, and also to submit changes to the database. The following code snippet shows how our IrishTidesDataContext class inherits from the DataContext object in the System.Data.Linq namespace. In the default constructor we take in a connection string, which is stored in the web.config file in the presentation layer and passed down, to the business logic layer and on to the IrishTidesDataContext. The main benefit of storing the database connection string at the presentation layer is that if we needed to change the database server it would just involve a simple change in the web.config, or if we wanted to run some unit tests we could pass in a connection string for a test database from our unit test project. The IrishTidesDataContext class will also contain individual methods for each of the stored procedures to be called, the function attribute is used to define the name of the stored procedure and the ExecuteMethodCall method of DataContext then knows how to call this stored procedure and return it's result. The previous code snippet for the TideGaugeQuery class called the GetTideGaugeById method within the IrishTidesDataContext class, in the code snippet below you can see how IrishTidesDataContext calls the stored procedure "dbo.GetTideGaugeById" passing in the id parameter to return a TideGaugeEntity object as a result.

```
public class IrishTidesDataContext : DataContext
{
    public IrishTidesDataContext(string connString)
            : base(connString, mappingSource)
    {
    }

    [Function(Name = "dbo.GetTideGaugeById")]
    public ISingleResult<TideGaugeEntity>
        GetTideGaugeById([Parameter(Name = "TideGaugeID",
        DbType = "BigInt")] System.Nullable<long> tideGaugeID)
    {
        IExecuteResult result = this.ExecuteMethodCall(this,
            ((MethodInfo)(MethodInfo.GetCurrentMethod())),
            tideGaugeID);
        return ((ISingleResult<TideGaugeEntity>)
            (result.ReturnValue));
    }
```

Database Layer

The database layer contains the RDBMS and the database itself. The DBMS for this Irish Tides project is Microsoft SQL Server 2005. *Figure 8 General architecture of SQL Server, Delaney* (2007 depicts the general architecture of SQL Server 2005 and it's four major components, which are protocols, query processor (relational engine), storage engine, and the SQLOS. According to Delaney (2007) it is the protocols layer, which first receives a request and *"translates it into a form that the relational engine can work with, and it also takes the final results of any queries, status messages, or error messages and translates them* *into a form the client can understand before sending them back to the client. The relational engine layer accepts SQL batches and determines what to do with them. For Transact-SQL queries and programming constructs, it parses, compiles, and optimizes the request and oversees the process of executing the batch. As the batch is executed, if data is needed, a request for that data is passed to the storage engine. The storage engine manages all data access, both through transaction-based commands and bulk operations such as backup, bulk insert, and certain DBCC (Database Consistency Checker) commands. The SQLOS layer handles activities that are normally considered to be operating system responsibilities, such as thread management (scheduling), synchronization primitives, deadlock detection, and memory management, including the buffer pool.".*

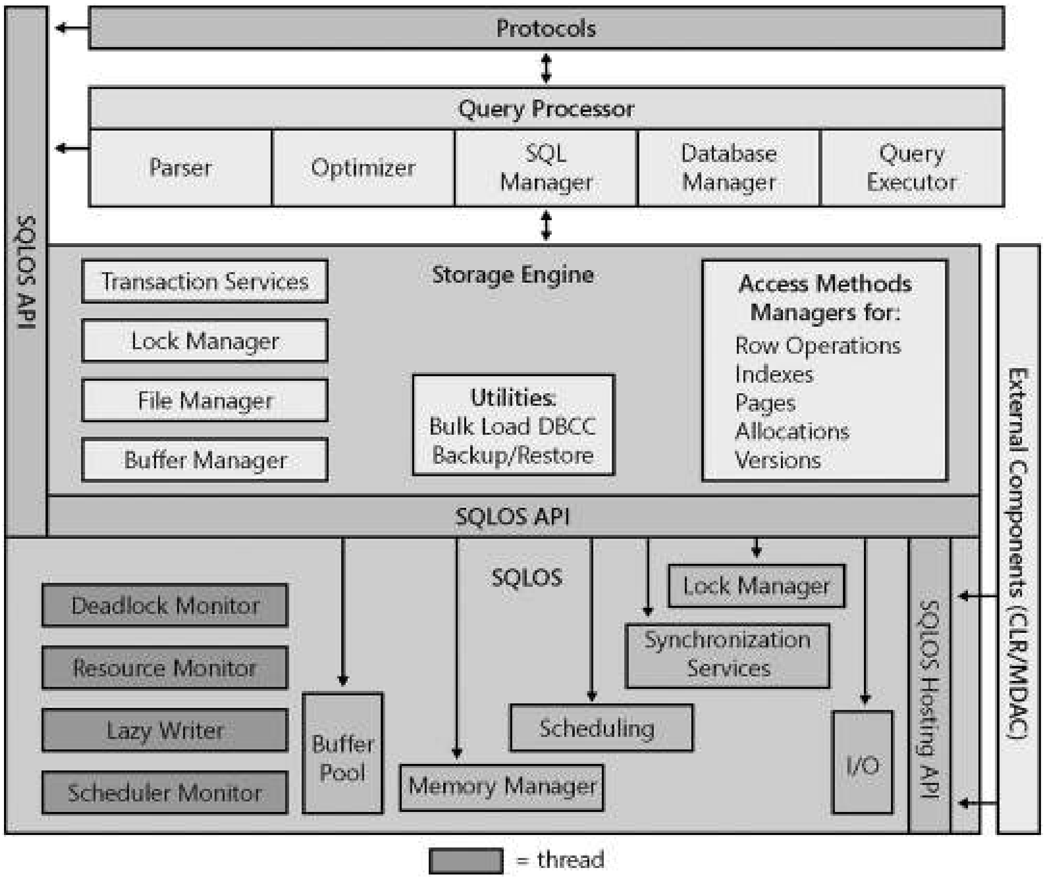


Figure 8 General architecture of SQL Server, Delaney (2007)

Chapter 4 – 3rd Party Controls

Graphing Component

My main aim was to find a free, preferably open-source graphing library for C# that would render the graph as an image such as a PNG or JPG. The Marine Institute currently use two different graphing components, crystal reports and another flash based library. The crystal reports requires a license and an executable to be installed on the web server, the flash based graphs require a client plugin, and this was not compatible with some mobile phones and PDA's which were only capable of viewing standard images. I drafted a shortlist of the following .NET graphing libraries and generated some sample graphs with each of the libraries. The resulting graphs were then matched against the Marine Institute's requirements to determine which library would be most suitable.

Selecting a Graphing Components

I tested the following .NET graphing libraries, ZedGraph, Nplot, WebChart and Google Charts and have provided some background information on each of these.

ZedGraph

Website:	http://zedgraph.org/
License:	Library General Public License
Opensource:	Yes

ZedGraph is a free open source C# graphing library that is maintained by SourceForge and is licensed under the LGPL (Library General Public License). It contains a set of classes for creating Line & Symbol Charts, Pie Charts, and Bar Charts among others and contains both an ASP.NET Web Control and a Windows Forms Control, and according to the ZedGraphWiki (2007) *"The classes provide a high degree of flexibility -- almost every* *aspect of the graph can be user-modified. At the same time, usage of the classes is kept simple by providing default values for all of the graph attributes. The classes include code for choosing appropriate scale ranges and step sizes based on the range of data values being plotted"*. The ZedGraph Wiki provided some excellent sample graphs to get me started and the documentation was also very useful.

Nplot

Website: http://netcontrols.org/nplot/wiki/

License: 3-clause-BSD license

Opensource: Yes

Nplot, which is formerly known as ScPL (Scientific Plotting Library), is another free charting library for .NET that is also maintained by SourceForge, but is licensed under the terms of a 3-clause-BSD (Berkeley Software Distribution) license. It offers pretty much the same range of graphs as ZedGraph does, and would appear to have been designed with scientific data in mind. It claims to boast an elegant and flexible API that includes controls for windows forms, ASP.NET and a class for creating Bitmaps, however very few code examples are provided on NPlot's Wiki, and the tutorials and documentation are poor in comparison to other libraries.

Google Chart API

Website: http://code.google.com/p/googlechartsharp/

License: None

Opensource: No

The Google Chart API lets you dynamically generate charts as a PNG-format image in response to a URL. Several types of image can be generated, including line, bar, and pie charts, venn diagrams and scatter plots. For each image type, you can specify attributes such as size, colors, and labels. GoogleChartSharp is a C# wrapper for the Chart API. Although the quality of the graphs produced are of excellent quality and the API is very simple to use, the Google chart API is not a customisable as some of the other libraries and so the graphs tend to be very basic.

WebChart

Website:	http://www.carlosag.net/Tools/WebChart/
License:	None
Opensource:	No

WebControl is yet another free ASP.NET library for creating charts, that renders as images such as PNG, JPG, and GIF. The API supports Line Charts, Column Charts, Area Charts, Scattered Charts, and Pie Charts. The WebChart web site contains some sample charts in both C# and VB.NET and also some class level documentation.

Final Selection

The graphing component needed to support the following MI requirements:

- Plotting of scatter graphs with an X-Axis displaying time in HH:mm, and a Y-Axis displaying numeric data to two decimal places.
- The Y-Axis should also support negative values and in such case the labels of the Y-axis should appear outside the outer perimeter of the graph and not at the zero line level within the graph.
- It should be possible to supply the graphing component with a collection object containing data points based on the selected date range and tide gauge instrument, and have it dynamically render the graph during page load.

- The MI also has a requirement that the graph can be rendered in all of the following 3 styles, White background with Blue curve, Blue background with White curve, and Blue background and Yellow curve

At first glance I would have taught that Nplot would be the most suitable to the needs of my project as it seems to have been designed with scientific data in mind, however in my opinion it lacks documentation and samples and there would seem to be a steep learning curve associated with it. I found it difficult to dynamically build my data point lists, and the lack of examples and documentation meant I struggled to get the graphs to visually appear like any of the 3 predefined styles required by the MI. The Google Chart API and GoogleChartSharp wrapper are extremely basic and image size is also restricted. Another potential problem with the Google API is because we are only linking to Google's site and have no actual library to work with if Google were to stop supporting this API then my charting service would be finished.

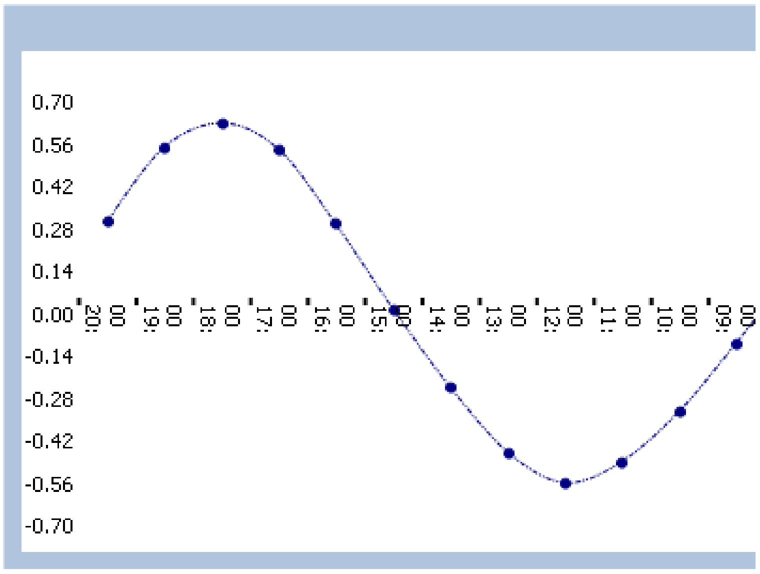


Figure 9 Nplot sample graph with X-Axis labels

The WebChart library generated some really good line graphs but I encountered issues with the placement of x-axis labels when the y-axis used a range that contained values less than zero as shown in *Figure 9 Nplot sample graph with X-Axis labels*.

The ZedGraph included a project Wiki, documentation and some excellent sample graphs, the graphs were easy to generate and are fully customizable and can be rendered as an image. It was easy to dynamically build a data point list using the ZedGraph PointPairList collection and also to visually match all of the predefined graph styles outlined by the MI. Overall the ZedGraph control was the most suitable control to meet the requirements of the MI and was therefore selected as the 3rd party graphing library to be used within this project.

log4net

Website:	http://logging.apache.org/log4net//index.html
License:	Apache License, Version 2.0
Opensource:	Yes

log4net is a logging framework that is maintained by the Apache Software Foundation and has been ported from the log4j (logging for java) framework to the Microsoft .NET Framework. log4net is a tool to help the programmer output log statements to a variety of output targets such as console, file, database, and SMTP servers, and is used to trace problems that occur at application runtime. It is a proven robust and reliable architecture that was designed with two distinct goals in mind: speed and flexibility, so that log statements can remain in production code without incurring a high performance cost.

The log4net library is used by the tidal analyser windows service for debugging the code and also for monitoring the application during runtime. In the windows service log4net will record all logging statements to log files using the log4net RollingFileAppender, which can be configured via a log4net config file. The config file allows you can specify the type of appender to be used, the maximum file size of each log file, the number of files it will roll over too before it starts to overwrite files, and the pattern that will be used to log each statement. Below is the log4net config file used by the tidal analyser windows service.

```
<log4net>
    <appender name="RollingFile"
         type="log4net.Appender.RollingFileAppender">
        <file value="C:\IrishTides\TidalAnalyser\logs\tidal.log" />
        <appendToFile value="true" />
        <maximumFileSize value="5MB" />
        <maxSizeRollBackups value="2" />
        <layout type="log4net.Layout.PatternLayout">
            <conversionPattern value="%date [%thread] %-5level %logger -
            %message%newline" />
        </layout>
    </appender>
    <root>
        <level value="DEBUG" />
        <appender-ref ref="RollingFile" />
    </root>
</log4net>
```

The config file then needs to be configured in the application before the logger can used and can be done so using the following statements:

```
private static readonly ILog log = LogManager.GetLogger(
      MethodBase.GetCurrentMethod().DeclaringType);
log4net.Config.XmlConfigurator.Configure(new FileInfo(@ConfigFile));
```

log4net can also log messages at the following logging levels FATAL, ERROR, WARN, INFO, DEBUG, and TRACE. For instance during startup the windows service use the following statement to log it’s startup event.

```
log.Debug("-> TidalAnalyserService: Started");
```

AjaxControlToolkit

Website:	http://www.asp.net/ajax/default.aspx
License:	Microsoft Public License (Ms-PL)
Opensource:	Yes

The ASP.NET Ajax Control Toolkit is an open source toolkit from Microsoft and the ASP.NET AJAX community, which provides a rich suite of web controls that can be used to develop interactive web pages. It is built on top of the Microsoft ASP.NET AJAX framework, and provides a powerful infrastructure to write reusable, customizable and extensible ASP.NET AJAX extenders and controls. The toolkit also comes with a great set of example web pages for all of the toolkits controls.

Controls from the Ajax Control Toolkit that were used by the web layer of the Irish Tides project include the UpdatePanel, CalendarExtender, MaskedEditValidator, and the MaskedEditExtender.

The UpdatePanel allows you to refresh selected parts of the web page instead of having to refresh the whole page and does so without the need to write any client side scripts. This partial-page updating or rendering is achieved by performing an asynchronous postback, from which only the HTML for the region of the page wrapped by the control is sent by the server asynchronously through an Ajax request. An asynchronous postback can be activated using triggers identified within the UpdatePanel control.

The CalendarExtender can be attached to any ASP.NET TextBox control to provide client-side date-picking functionality with customizable date format and UI in a popup control. The user can then interact with the calendar by clicking on a day, navigate to months and years by clicking on the left and right arrows and perform other such actions without a postback.

Like the CalendarExtender the MaskedEditValidator must also be attached to a TextBox and is used to restrict the kind of text that can be entered in the TextBox. It applies a mask to the user input that allows specific character formats such as numeric, date, time, and datetime to be entered. To validate the input the MaskedEditValidator must be used in conjunction with a MaskedEditExtender.

Chapter 5 – The Application

This chapter provides and overview of the Irish Tides web application and Tidal Analyser Service and how each of the user stories described in the *Irish Tides User Stories* section have been solved in the application.

Web Logon

For the Web Logon user story I identified the following tasks in the project estimates section (see *Estimates*).

1. *Create web form for user logon, log-off pages.*

   The logon page (Logon.aspx) as shown in *Figure 10 Logon Page* is the entry point of the web application, it is an ASPX web page with TextBox controls for Username and Password and a Logon Button, and all users must be authenticated before they can gain access to the web application. Users that supply a valid Username and Password (which will be verified against the web users table in the database) will be redirected to the default page within the website and a session will be created for that user, an error message is displayed to the user if they enter an incorrect combination of Username and Password.
2. *Create the default page that the user will be brought to once they have been validated.*

   Once logged in successfully the user will be brought to the default-authenticated page as shown in *Figure 11 Default Authenticated Page*. The area in the bottom right within the red lines is the area each page within the authenticated section that is configurable, everything outside of the red lines is maintained by the master page. The default page will display a welcome greeting to the user.

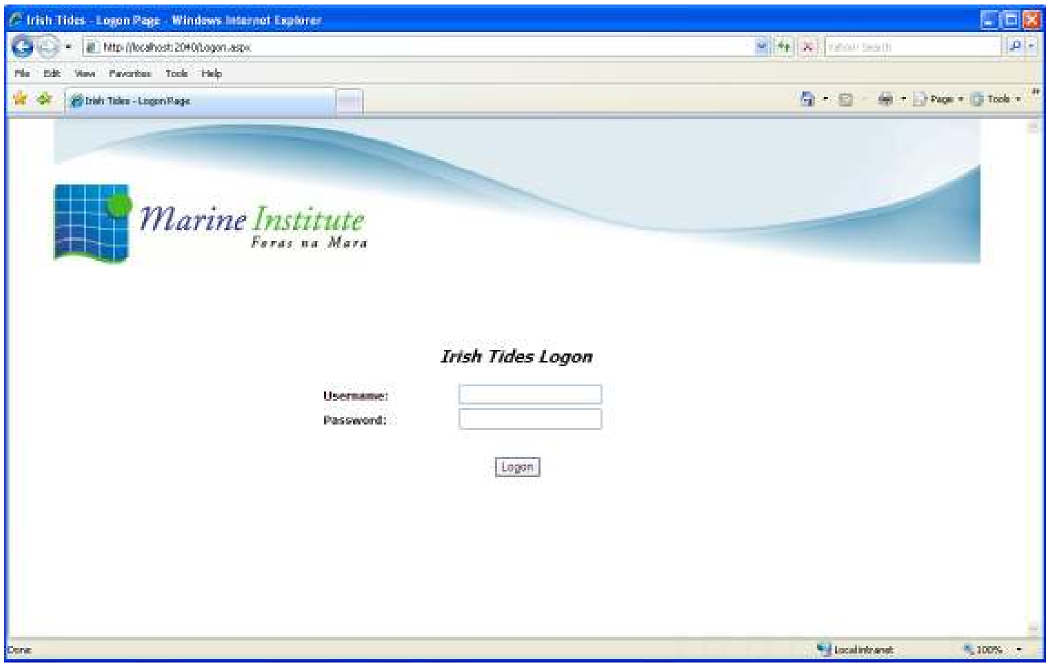

Irish Tides - Logon Page - Windows Internet Explorer
http://localhost:2040/Logon.aspx
File Edit View Favorites Tools Help
Irish Tides - Logon Page
Marine Institute
Foras na Mara
Irish Tides Logon
Username:
Password:
Logon
Done
Local intranet
100%

Figure 10 Logon Page

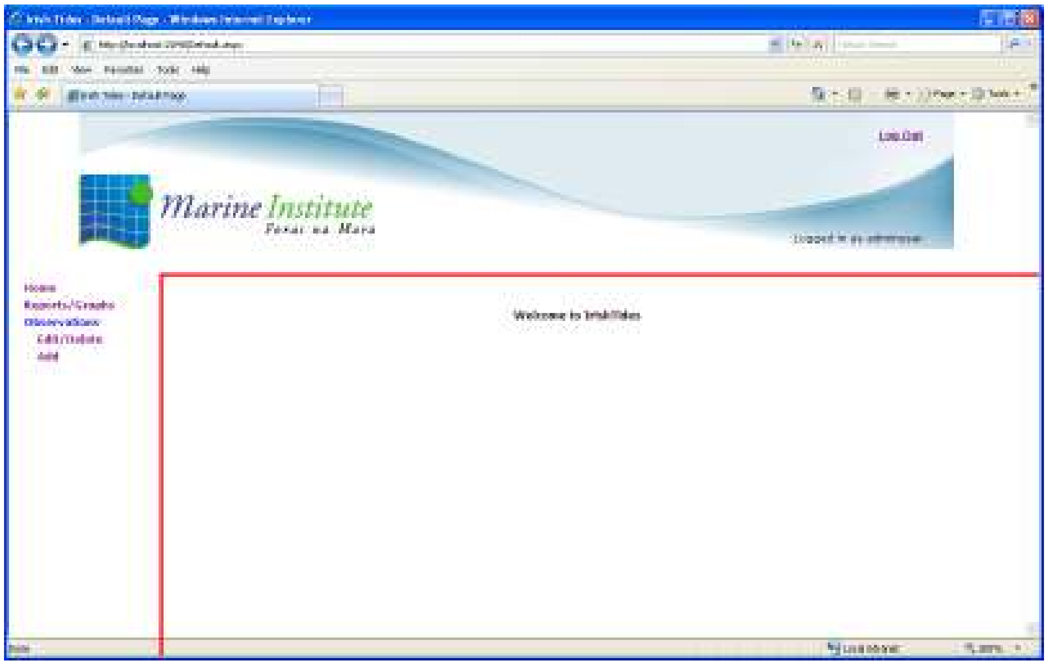

Irish Tides - Default Page - Windows Internet Explorer
File Edit View Favorites Tools Help
Irish Tides - Default Page
Log Out
Marine Institute
Foras na Mara
Logged in as administrator
Home
Reports/Graphs
Observations
Edit/Delete
Add
Welcome to IrishTides

Figure 11 Default Authenticated Page

3. *Create a master page to be used on all pages once the user has been validated.*

   A Master Page is used for all pages with the authentication section of the web site. As you can see in *Figure 11 Default Authenticated Page* everything outside of the red box is the controlled by the master page, so the navigation bar on the left and the header, which contains a link to 'Log Out' of the application and displays the username of the person logged in is all contained within the master page and will therefore appear on all other pages that use the master page.

Data Reporting

For the Data Reporting user story I identified the following tasks in the project estimates section (see *Estimates*).

1. *Develop a web form to allow users to specify criteria for the report they wish to generate.* By selecting the Reports/Graphs link in the navigation bar on the left hand side of the screen (see *Figure 11 Default Authenticated Page*) will bring you to the Custom Reports & Graphs Page (CustomReports.aspx) as shown in *Figure 12 Custom Reports (a)*. Here the user can select the Tide Gauge, Instrument, and Date Range and the report type they wish to generate. When the user clicks the 'View Report' button a client side JavaScript function located in the master page is invoked that will generate a parameterised URI based on what the user has selected, and will open this URI in a new window.

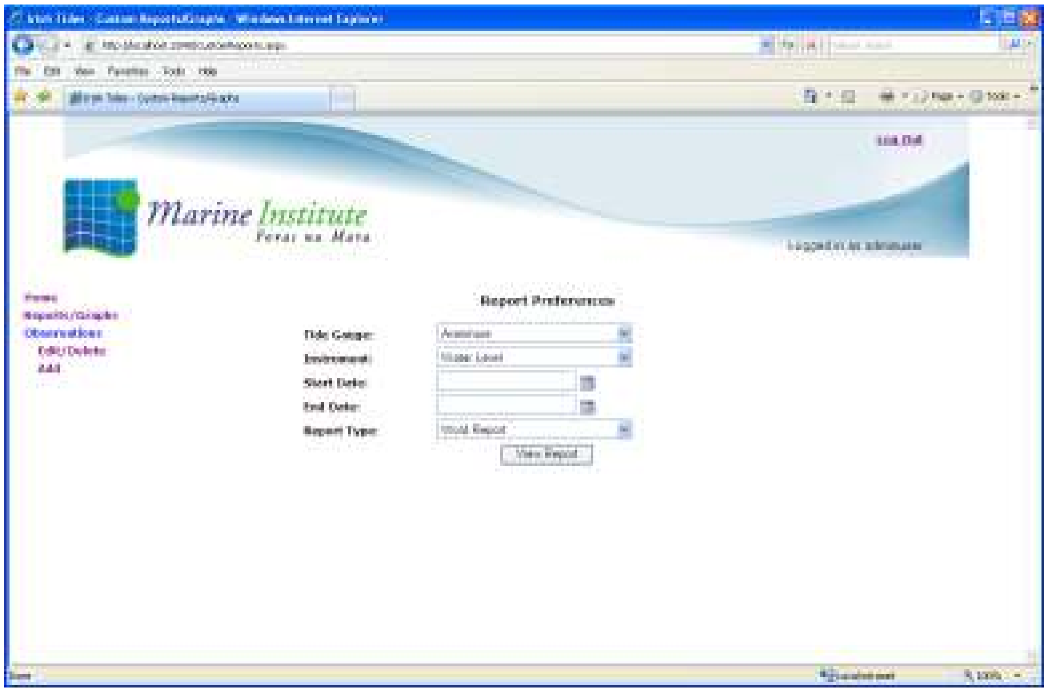
   The two data reports available are Word Report and CSV (Excel) Report. When a Word Report is selected a JavaScript function similar to the following will be called which will open the WordReportViewer.aspx page in a new window:

   ```
   window.open('/reports/WordReportViewer.aspx?tideGaugeId=1&instrumentId=1&startDate
   =04/08/2009&endDate=05/08/2009&title=Ballyglass%20-%20Water%20Level', '',
   'toolbar=0,scrollbars=0,location=0,statusbar=0,menubar=0');
   ```

   For a CSV (Excel) Report the function would instead open the CSVReportViewer.apsx page in a new window with a URI similar to the following:

   ```
   window.open('/reports/CSVReportViewer.aspx?tideGaugeId=1&instrumentId=1&startDate=
   04/08/2009&endDate=05/08/2009&title=Ballyglass%20-%20Water%20Level ', '',
   'toolbar=0,scrollbars=0,location=0,statusbar=0,menubar=0');
   ```

The third option in the 'Report Type' drop down is a Graph, which will be explained in the section *Graph Reporting*.



Report Preferences

Tide Gauge:	[illegible]
Instrument:	Water Level
Start Date:	
End Date:	
Report Type:	Word Report

View Report

Figure 12 Custom Reports (a)

2. *Develop a web form to generate Microsoft Word reports.*

   WordReportViewer.aspx is the page that is rendered to the user in a new window when a 'Word Report' is selected. It will not be displayed to the user as a standard web page, instead the response content type of the page is set to `"application/msword"` and therefore provided the client has Microsoft Word installed the window will be displayed to the user as a Word document instead of a web page. The ASPX presentation part of the page contains an ASP.NET DataList control which is used to organize the columns and the style of the report the user will see, the WordReportViewer.aspx.cs code behind class will parse the parameter list passed in via the URL i.e. WordReportViewer.aspx?**tideGaugeId**=1&**instrumentId**=1&**startDate**=04/08/2009&**endDate**=05/08/2009&**title**=Ballyglass%20-%20Water%20Level, and extract the value for tideGaugeId, instrumentId, startDate, endDate and title. These values are then used to query the database and return a List object, which will be assigned as a DataSource to the DataList control and displayed to the user as table in the word document as shown in *Figure 21 Irish Tides Microsoft Word Report*.

3. *Develop a web form to generate Microsoft Excel reports.*

CSVReportViewer.aspx is the page that is rendered to the user in a new window when a 'CSV Report' (Excel Report) is selected. Like with the word document the CSV report will not be displayed to the user as a standard web page, instead the response content type of this page is set to `"application/x-msexcel"` and therefore provided the client has Microsoft Excel installed the window will be displayed to the user as a excel document instead of a web page. Unlike in the word document there is no need for the ASPX presentation part of the page to contain anything, instead we simply assign a string of comma separated values to the HTTP response output stream like the following code snippet shows:

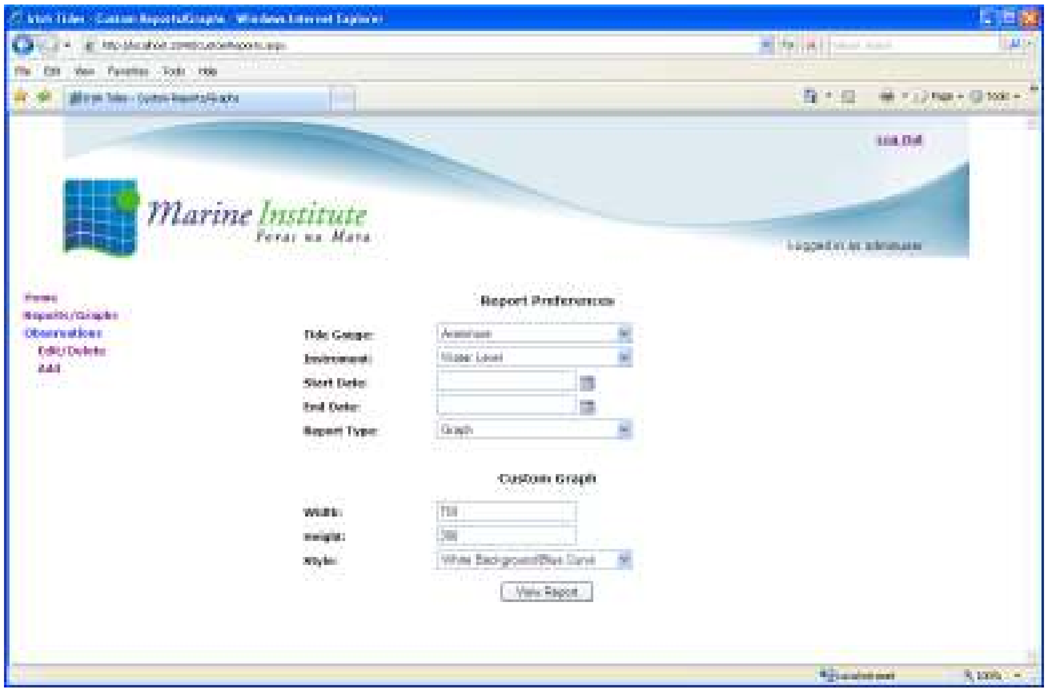
```
Response.Write("TideGuage, Date, Water Level\r\nBallyglass, 04/08/2009
12:00:00, -1.46700000\r\nBallyglass, 03/08/2009 11:54:00, -
1.41100000\r\n");
```

Like in the word document we also parse the URL for parameters which are then used to query the database and return a List object, which is then used to build a string of comma separated values with the string `"\r\n"` representing a new row in the Excel spreadsheet, a example of which can be seen in *Figure 22 Irish Tides Microsoft Excel Report*.

Graph Reporting

The following Graph Reporting tasks were identified in the *Estimates* section.

1. *Amend the data reports web form to allow for graph reports.*



Report Preferences

Tide Gauge:	[illegible]
Environment:	Water Level
Start Date:	
End Date:	
Report Type:	Graph

Custom Graph

Width:	[illegible]
Height:	[illegible]
Style:	White Background/Blue Curve

Figure 13 Custom Reports (b)

An additional 'Graph' item was added to the 'Report Type' DropDownList, and the DropDownList was placed within the AjaxControlToolkit's UpdatePanel control. The UpdatePanel control permits the DropDownList to make asynchronous postbacks so that when the 'Graph' item is selected in the DropDownList, a 'Custom Graph' panel containing width, height, and style controls will be made visible without the need for a full page refresh as shown in *Figure 13 Custom Reports (b)*. By adding a trigger for the DropDownList control to the UpdatePanel, we are able to configure an asynchronous postback as shown in the code snippet below.

```
<Triggers>
  <asp:AsyncPostBackTrigger ControlID="ddlReportType"
      EventName="SelectedIndexChanged" />
</Triggers>
```

Client side JavaScript is again used on this page so when a user selects 'View Report' a new window will open the ChartGenerator.aspx page with a URI similar to the following

```
window.open('/ChartViewer.aspx?tideGaugeId=1&instrumentId=1&startDate=04/08/2009&
endDate=05/08/2009&title=Ballyglass%20-
%20Water%20Level&height=300&width=750&graphStyle=1', '',
'toolbar=0,scrollbars=0,location=0,statusbar=0,menubar=0');
```

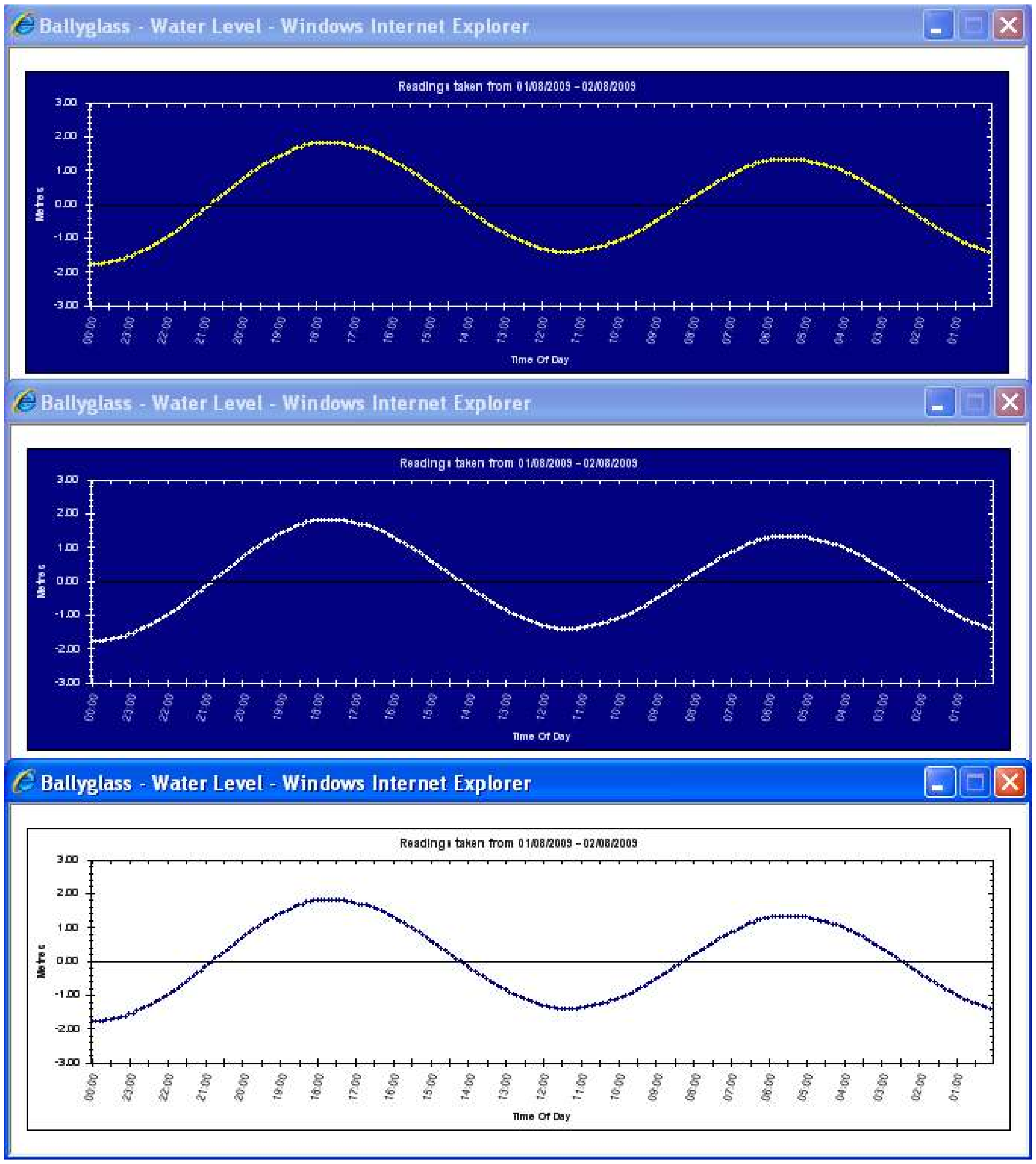


Figure 14 All Predefined Graph Styles

2. *Develop a web form to generate tidal graphs as images.*

   The ChartGenerator.aspx page uses the ZedGraph library to generate charts. A ZedGraphWeb control is placed on the presentation part of the ASPX page as shown in the code snippet below.

```
<cc1:zedgraphweb id="IrishTidesGraph" runat="server"
  RenderMode="RawImage">
</cc1:zedgraphweb>
```

The code behind ChartGenerator.aspx.cs contains the ZedGraph code to dynamically build a graph based on the parameters passed in via the URI. Along with the now standard parameters for tideGaugeId, instrumentId, startDate, endDate and title, ChartGenerator.aspx also contains width, height and graphStyle parameters. The code behind class uses these parameters to generate the customised graph. Currently only three types of graph styles will be supported, and they are shown in *Figure 14 All Predefined Graph Styles*.

Observations Add New

The following tasks were identified in the *Estimates* section for Observations Edit/Delete functionality.

1. *Develop a web form to enable users to add new observations.*

   I created a new ASPX web form AddObservation.aspx, which can be navigated to from the side menu under Observations -> Add. Occasionally users will need to manually add a new observation item and this form will allow them to do so. The from contains some standard ASP.NET web controls like a DropDownList for the DataSource, TextBox's for date, reading and comments, and a Button for 'Add Record'. The right hand side of the date TextBox contains an AjaxControlToolkit Calendar Control which when selected gives the user a calendar date picker. Two additional AjaxControlToolkit controls namely the MaskedEditValidator and MaskedEditExtender are used to validate that the date and reading TextBox's contain text, and also that the text conforms to an allowable format, i.e. the date time format 'dd/MM/yyyy HH:mm' for the date TextBox, and a decimal format for the reading TextBox. Provided that the form can be validated the user will be able to select 'Add Record' and have the observation item added to the database, if the form can't be validated an error message will be displayed to the user in a Label control.

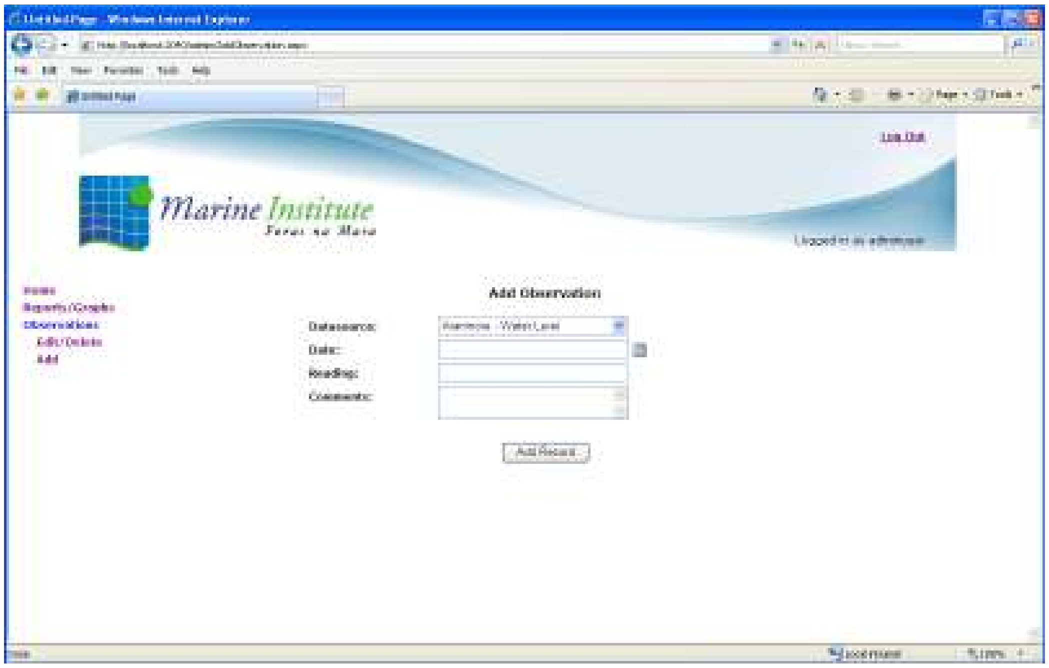


Figure 15 Add Observations

2. *Create/Update Stored Procedures for adding new observations.*

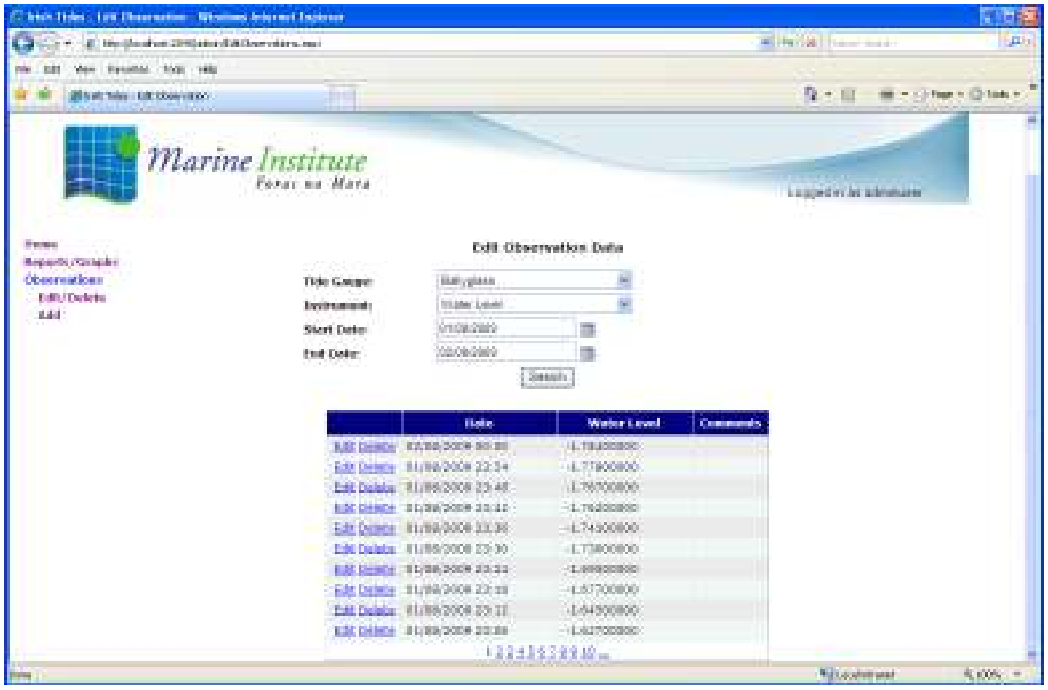
   Three stored procedures had to be created for adding observation records to the database, one for each of the instrument types of water level, pressure and temperature. This stored procedure will then be called by a function in the IrishTidesDataContext.cs class in the IrishTidel.Dal layer.

Observations Edit/Delete

The following tasks were identified in the *Estimates* section for Observations Edit/Delete functionality.

1. *Develop a web form to enable users to edit or delete observations.*

   The EditObservations.aspx page allows the user to edit observation records, or to remove them entirely from the database, and it can be navigated to from the side menu under Observations -> Edit/Delete. The user must first search for available observation records by selecting a ‘Tide Gauge’, Instrument, and a ‘Start Date’ and ‘End Date’ date range. The ‘Tide Gauge’ and Instrument DropDownLists are populated with values from the TideGauge and Instrument database tables respectively. The start and end date TextBox’s are again validated by the AjaxControlToolkit’s MaskedEditValidator and MaskedEditExtender controls.


Edit Observation Data

Tide Gauge:	Ballyglass
Instrument:	Water Level
Start Date:	01/08/2009
End Date:	02/08/2009

Search

	Date	Water Level	Comments
Edit Delete	02/08/2009 00:00	-1.78400000	
Edit Delete	01/08/2009 23:54	-1.77900000	
Edit Delete	01/08/2009 23:48	-1.76700000	
Edit Delete	01/08/2009 23:42	-1.76200000	
Edit Delete	01/08/2009 23:36	-1.74500000	
Edit Delete	01/08/2009 23:30	-1.73800000	
Edit Delete	01/08/2009 23:24	-1.69800000	
Edit Delete	01/08/2009 23:18	-1.67700000	
Edit Delete	01/08/2009 23:12	-1.64500000	
Edit Delete	01/08/2009 23:06	-1.62700000	

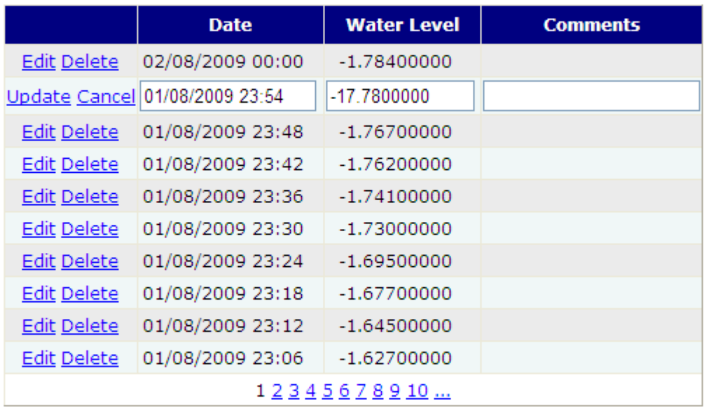
1 2 3 4 5 6 7 8 9 10 ...

Figure 16 Edit Observation Form

A list of observation records that match the search criteria will then be displayed to the user in tabular form using the ASP.NET GridView control once the user has selected the ‘Search’ button, as shown in *Figure 16 Edit Observation Form*. Users can also page through the records by selecting the page numbers beneath the search results table. By selecting ‘Edit’ on any of the observation records in the table will display editable TextBox fields to the user and they will then have the opportunity to make amendments to the TextBox fields before selecting ‘Update’ to commit the changes the database, or ‘Cancel’ to undo any changes, see *Figure 17 Update Observation Record*. Users can also select ‘Delete’ on any observation record to completely remove the observation item from the database.

2. *Create/Update Stored Procedures for edit and delete functionality.*

   A total of six stored procedures had to be created for updating and deleting observations records, as update and delete were both required for each of the instrument types of water level, pressure and temperature. These stored procedures will then be called by functions in the IrishTidesDataContext.cs class in the IrishTidel.Dal layer.



	Date	Water Level	Comments
Edit Delete	02/08/2009 00:00	-1.78400000	
Update Cancel	01/08/2009 23:54	-17.7800000	
Edit Delete	01/08/2009 23:48	-1.76700000	
Edit Delete	01/08/2009 23:42	-1.76200000	
Edit Delete	01/08/2009 23:36	-1.74100000	
Edit Delete	01/08/2009 23:30	-1.73000000	
Edit Delete	01/08/2009 23:24	-1.69500000	
Edit Delete	01/08/2009 23:18	-1.67700000	
Edit Delete	01/08/2009 23:12	-1.64500000	
Edit Delete	01/08/2009 23:06	-1.62700000	

1 2 3 4 5 6 7 8 9 10 ...

Figure 17 Update Observation Record

Detecting time gaps

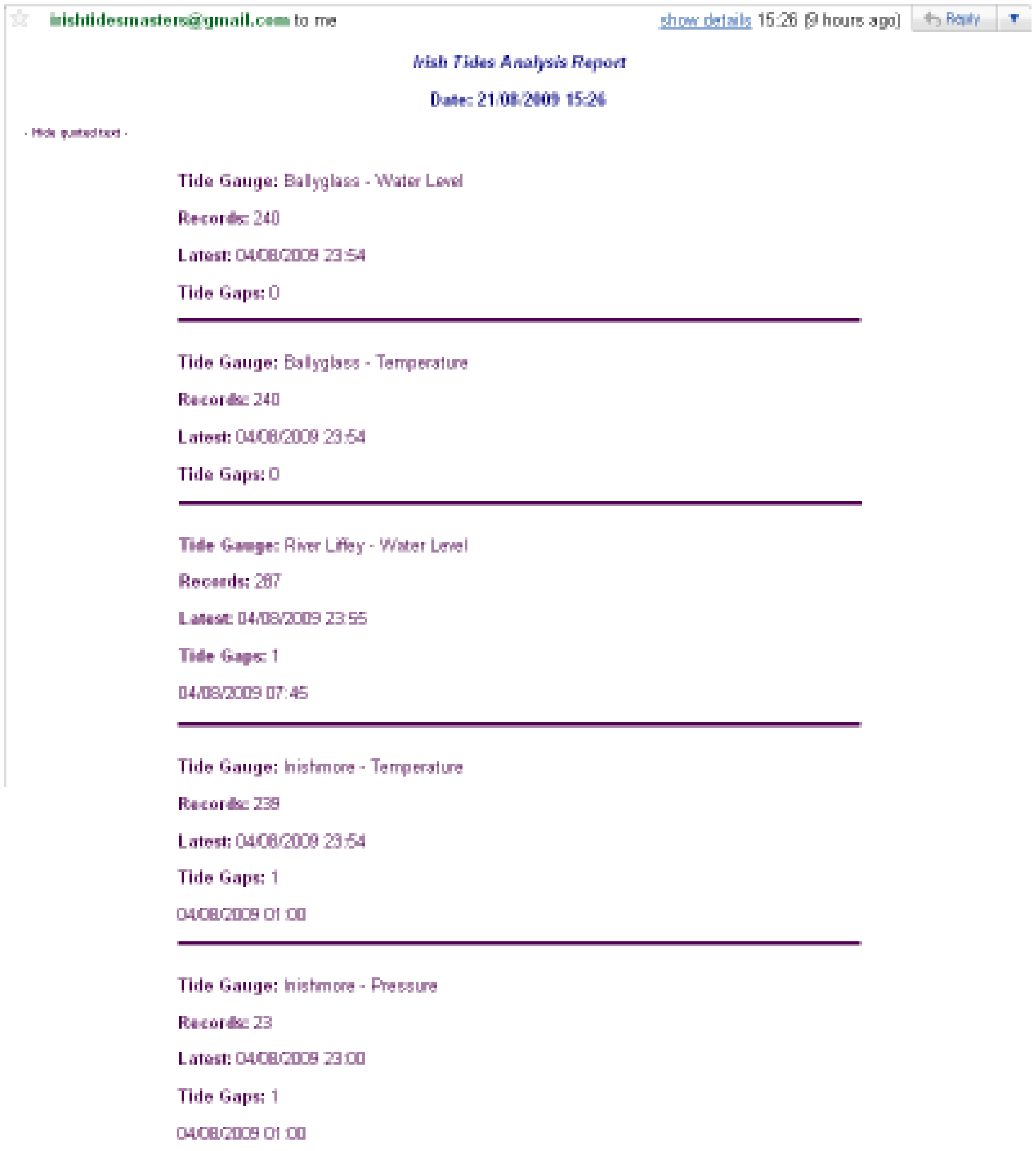
The following tasks were identified in the *Estimates* section for detecting time gaps.

1. *Develop windows service.*

   A windows service was created in the TidalAnalyser project within the Irish Tides visual studio solution, see *Tidal Analysis Service* for further details. The project contains an install.bat script within the config directory to install the executable as a windows service, and also as a TidalAnalysis.config file for configuring the log4net logging framework, which is used for monitoring the windows service during runtime.

2. *Implement functionality to detect time gaps.*

Due to the dynamic nature in which the time interval between observation reading's can be changed at any one of the tide gauge stations, it is not possible to make assumptions on the time differences betweens readings as this will vary across tide gauges and can be updated at any time. Therefore there was a need to create a function that could detect the expected time gap for each of the tide gauge stations based on the observation intervals recorded during a 24-hour period, and from this determine any time gaps that would appear to be missing.


irishtidesmasters@gmail.com to me

show details 15:26 (9 hours ago)

Irish Tides Analysis Report

Date: 21/08/2009 15:26

- Hide quoted text -

Tide Gauge: Ballyglass - Water Level
Records: 240
Latest: 04/08/2009 23:54
Tide Gaps: 0

Tide Gauge: Ballyglass - Temperature
Records: 240
Latest: 04/08/2009 23:54
Tide Gaps: 0

Tide Gauge: River Liffey - Water Level
Records: 287
Latest: 04/08/2009 23:55
Tide Gaps: 1
04/08/2009 07:45

Tide Gauge: Inishmore - Temperature
Records: 239
Latest: 04/08/2009 23:54
Tide Gaps: 1
04/08/2009 01:00

Tide Gauge: Inishmore - Pressure
Records: 23
Latest: 04/08/2009 23:00
Tide Gaps: 1
04/08/2009 01:00

Figure 18 Sample Time Gaps Analysis Report

3. *Add functionality to send email reports containing the time gaps to the OSS users.*

   The EmailManager.cs class has responsibility for the creation and sending of all emails. It reads the SenderEmailAddress, SenderEmailPassword, SmtpHost, and SmtpPort values from the App.config configuration file contained within the TidalAnalyser project and constructs a HTML based report, which will be sent to the recipients. *Figure 18 Sample Time Gaps Analysis Report* is a sample of the email reports that are sent after the time gaps analysis engine has analysed the dataset.

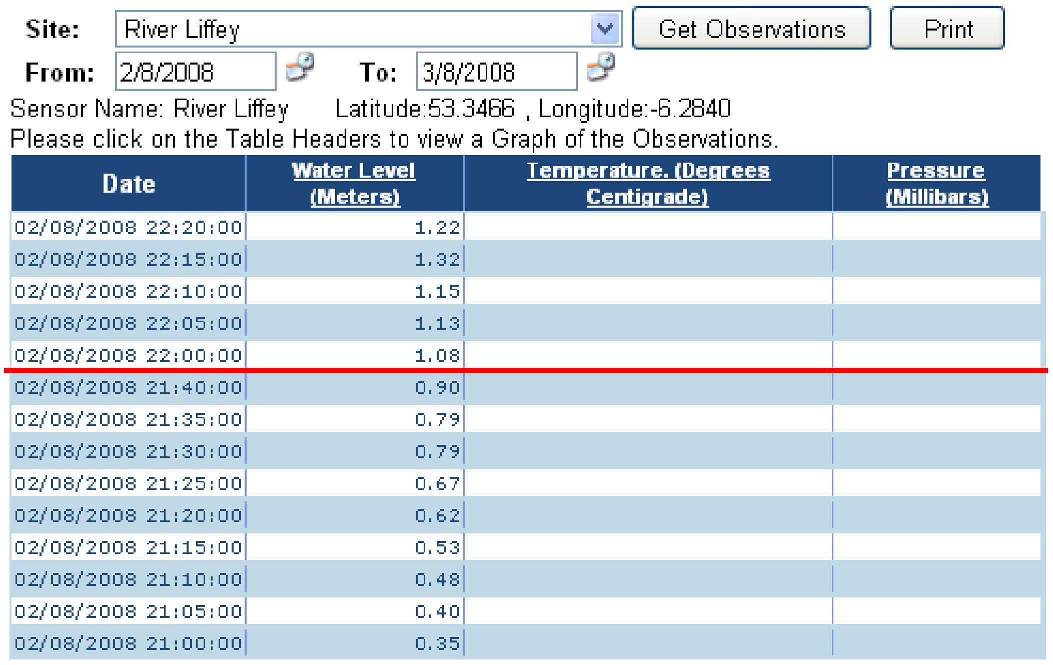
Chapter 6 – Evaluation

This chapter deals with the evaluation of the project outlined in the previous chapters, against what existed previously. In my thesis statement I said that I planned on *"Implementing functionality to identify and fix time gaps, and generate custom reports and graphs will improve the analysis, modification and reporting off tidal data in Ireland"*. I will now address each of the three areas of analysis, modification and reporting, and demonstrate how each of these areas has been improved upon.

How Analysis been improved?

As mentioned in the introduction chapter, the OSS department within the MI would like to know about time gaps in the data within a short time period of the data being collected. Time gaps can be caused by the likes of instrument malfunctioning on the tide gauges, or communication problems in transmission between the tide gauge and the base station.

Previously the only way of detecting if time gaps appeared in the tidal data was to manually scroll through the tidal observations that are exported to the MI's public website, or sometimes visitors to the site would report on any time gaps they encountered. The observations contain a list of all tide gauge readings for a 24-hour period for each of the three instruments, water level, temperature and pressure. Each tide gauge also has its own observation page, and so it was a tedious, error prone, and time-consuming task to check the observations for each of the tide gauges, and you were also reliant on the naked eye to spot any time gaps in the data. Water level is also generally recorded at more regular time intervals than pressure or temperature, most tide gauges record water level every 6 minutes, whereas temperature and pressure may only be recorded every hour, and so these varying time intervals across tide gauges would only add to the difficulty in manually detecting the time gaps accurately. *Figure 19 Irish Tides Observation with time gap* shows an observation page for the River Liffey tide gauge and highlighted by a red line is an example of time gaps that appear in this dataset between 21:40 and 22:00. The water level instrument on the River Liffey should be recording readings every 5 minutes but in this example is missing the 21:45, 21:50 and 21:55 readings.



Sensor Name: River Liffey Latitude:53.3466 , Longitude:-6.2840
Please click on the Table Headers to view a Graph of the Observations.

Date	Water Level (Meters)	Temperature. (Degrees Centigrade)	Pressure (Millibars)
02/08/2008 22:20:00	1.22		
02/08/2008 22:15:00	1.32		
02/08/2008 22:10:00	1.15		
02/08/2008 22:05:00	1.13		
02/08/2008 22:00:00	1.08		
02/08/2008 21:40:00	0.90		
02/08/2008 21:35:00	0.79		
02/08/2008 21:30:00	0.79		
02/08/2008 21:25:00	0.67		
02/08/2008 21:20:00	0.62		
02/08/2008 21:15:00	0.53		
02/08/2008 21:10:00	0.48		
02/08/2008 21:05:00	0.40		
02/08/2008 21:00:00	0.35		

Figure 19 Irish Tides Observation with time gap, Marine Institute (2009)

The analysis engine developed as part of this project will detect any time gaps in the dataset and send an email report to authorised users detailing each of the gaps. It has been developed as a windows service and will be set to run once daily, will check each of the active tide gauges and find the latest recorded reading for each of the instruments on the gauge. It will then analyse the data collected for the 24-hour period previous to this latest reading and generate a report, which will be sent as an email to all of the authorised report users. This is a huge improvement on the analysis of the Irish tide gauge data, as the OSS dept will now receive accurate daily reports outlining all of the time gaps detected in the tidal dataset for the previous 24-hours from there latest recorded reading.

How Modification has been improved?

Previously, members of the OSS dept had no way they could update tide gauge readings for any of the instruments on the tide gauges. Amendments to the tidal observations could only be made at the database table level itself. OSS members would not have the authorization or database knowledge to complete this task so therefore they would have to issue a database change request to the applications development team, which would then be queued to a developed to complete the change.

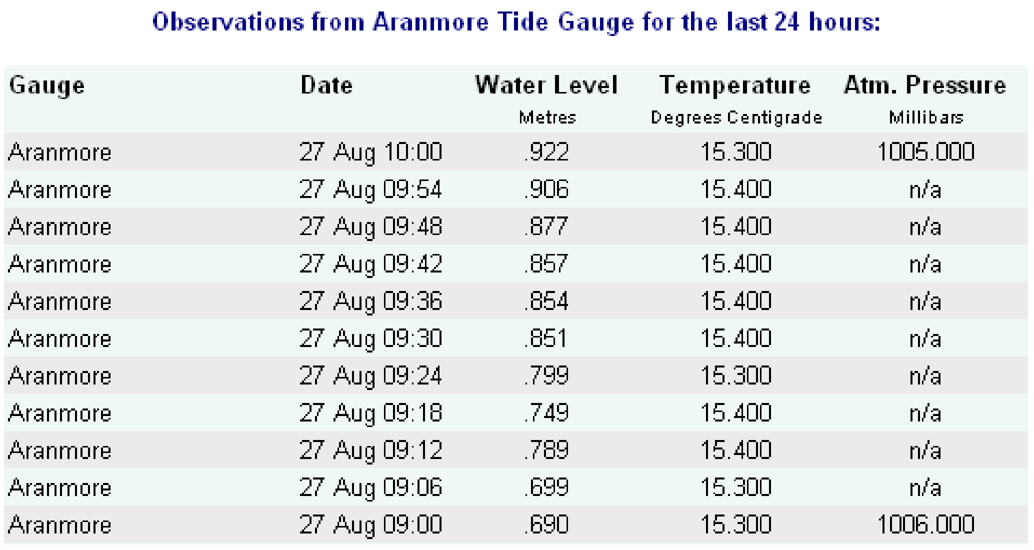
OSS had a requirement that authorised users from their team be able to make amendments to the tidal dataset to correct possible time gaps where possible. As part of this project I developed a series of web forms that allow users to search on a specific tide gauge instrument for a time period and amend any of the observation records within this dataset. Users can also add/remove individual observation records. Validation is also performed on these web forms before any database changes can take place.

The ability for OSS members to now be able to make amendments to the tidal data themselves and in real time is also a big improvement on the previous system. They no longer have to deal with the time consuming tasks of issuing database change requests or the significant time delays before the changes are reflected in the database as this can now all be done in real time.

How Reporting has been improved?

OSS wished to improve the reporting side of the Irish tides data also, and had requirements for a facility to generate data reports in both Microsoft Word and Excel format, and also to replace their existing tidal graphing components with a free ASP.NET library that could render the graphs as an Image.

Data Reports



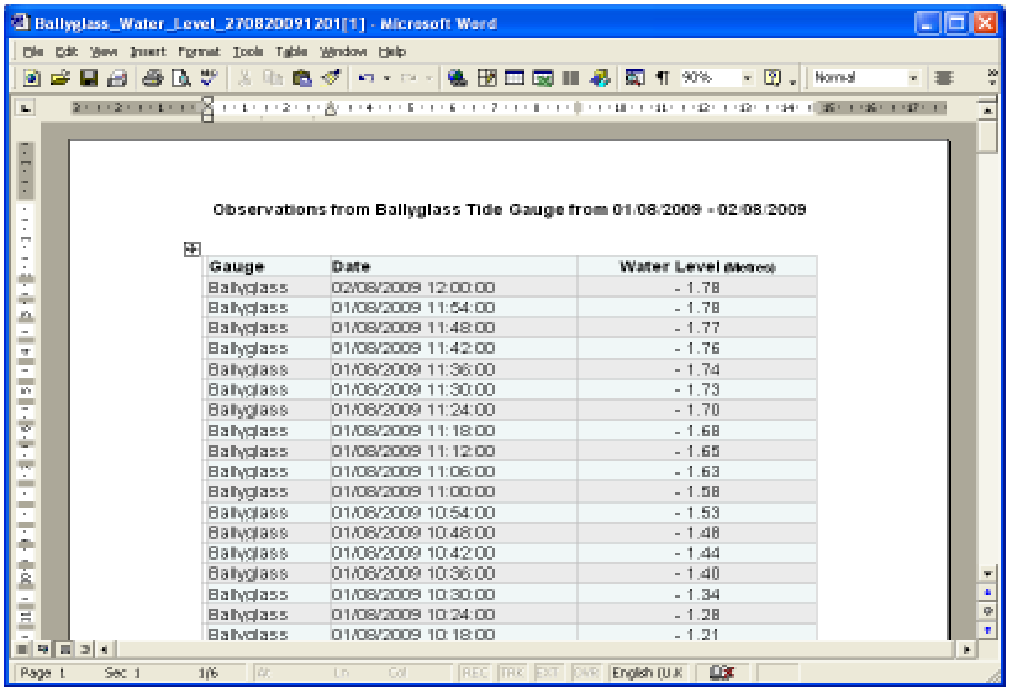
Observations from Aranmore Tide Gauge for the last 24 hours:

Gauge	Date	Water Level Metres	Temperature Degrees Centigrade	Atm. Pressure Millibars
Aranmore	27 Aug 10:00	.922	15.300	1005.000
Aranmore	27 Aug 09:54	.906	15.400	n/a
Aranmore	27 Aug 09:48	.877	15.400	n/a
Aranmore	27 Aug 09:42	.857	15.400	n/a
Aranmore	27 Aug 09:36	.854	15.400	n/a
Aranmore	27 Aug 09:30	.851	15.400	n/a
Aranmore	27 Aug 09:24	.799	15.300	n/a
Aranmore	27 Aug 09:18	.749	15.400	n/a
Aranmore	27 Aug 09:12	.789	15.400	n/a
Aranmore	27 Aug 09:06	.699	15.300	n/a
Aranmore	27 Aug 09:00	.690	15.300	1006.000

Figure 20 Irish Tides HTML Observations Report

The only data reporting that was previously available for Irish Tides was published to a HTML table on the www.marine.ie website as shown in *Figure 20 Irish Tides HTML Observations Report*. This provided valuable information to users of the website but OSS often required reports in other formats that could be exchanged over email.

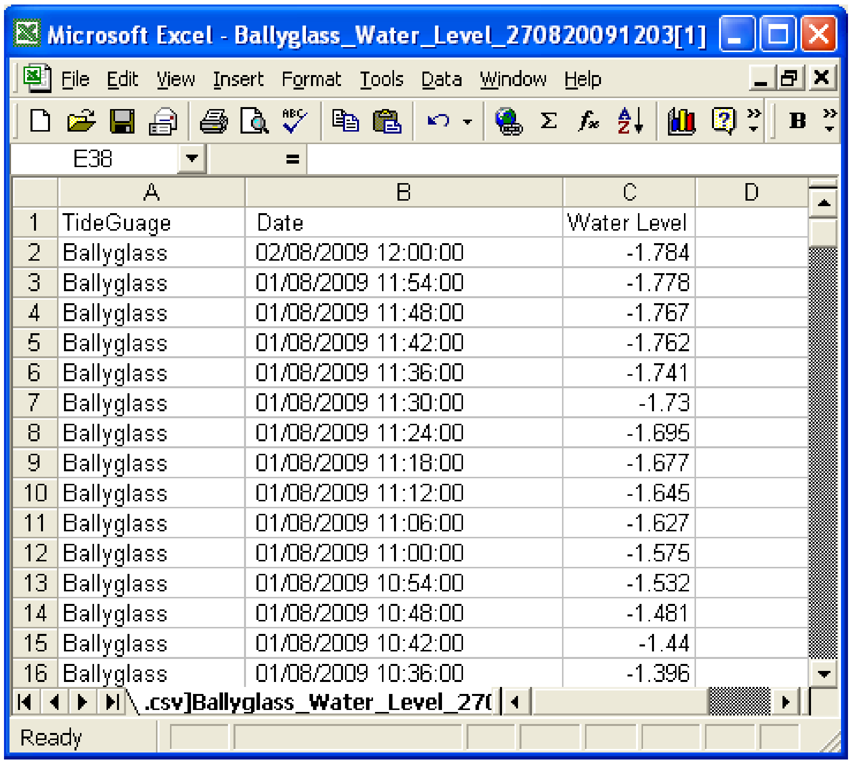
As part of this project I have developed a web form that provides a user-friendly interface to query the tidal data based on factors such as date range, tide gauge location and instrument, and users then have the option to have these reports exportable to either Microsoft Word (see *Figure 21 Irish Tides Microsoft Word Report*) or Excel (See *Figure 22 Irish Tides Microsoft Excel Report*) formats.


Ballyglass_Water_Level_270820091201[1] - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Page 1 Sec 1 1/6 English (U.K

Observations from Ballyglass Tide Gauge from 01/08/2009 - 02/08/2009

Gauge	Date	Water Level (Metres)
Ballyglass	02/08/2009 12:00:00	- 1.78
Ballyglass	01/08/2009 11:54:00	- 1.78
Ballyglass	01/08/2009 11:48:00	- 1.77
Ballyglass	01/08/2009 11:42:00	- 1.76
Ballyglass	01/08/2009 11:36:00	- 1.74
Ballyglass	01/08/2009 11:30:00	- 1.73
Ballyglass	01/08/2009 11:24:00	- 1.70
Ballyglass	01/08/2009 11:18:00	- 1.68
Ballyglass	01/08/2009 11:12:00	- 1.65
Ballyglass	01/08/2009 11:06:00	- 1.63
Ballyglass	01/08/2009 11:00:00	- 1.58
Ballyglass	01/08/2009 10:54:00	- 1.53
Ballyglass	01/08/2009 10:48:00	- 1.48
Ballyglass	01/08/2009 10:42:00	- 1.44
Ballyglass	01/08/2009 10:36:00	- 1.40
Ballyglass	01/08/2009 10:30:00	- 1.34
Ballyglass	01/08/2009 10:24:00	- 1.28
Ballyglass	01/08/2009 10:18:00	- 1.21

Figure 21 Irish Tides Microsoft Word Report


Microsoft Excel - Ballyglass_Water_Level_270820091203[1]
File Edit View Insert Format Tools Data Window Help
E38
=
.csv]Ballyglass_Water_Level_27(
Ready

	A	B	C	D
1	TideGuage	Date	Water Level	
2	Ballyglass	02/08/2009 12:00:00	-1.784	
3	Ballyglass	01/08/2009 11:54:00	-1.778	
4	Ballyglass	01/08/2009 11:48:00	-1.767	
5	Ballyglass	01/08/2009 11:42:00	-1.762	
6	Ballyglass	01/08/2009 11:36:00	-1.741	
7	Ballyglass	01/08/2009 11:30:00	-1.73	
8	Ballyglass	01/08/2009 11:24:00	-1.695	
9	Ballyglass	01/08/2009 11:18:00	-1.677	
10	Ballyglass	01/08/2009 11:12:00	-1.645	
11	Ballyglass	01/08/2009 11:06:00	-1.627	
12	Ballyglass	01/08/2009 11:00:00	-1.575	
13	Ballyglass	01/08/2009 10:54:00	-1.532	
14	Ballyglass	01/08/2009 10:48:00	-1.481	
15	Ballyglass	01/08/2009 10:42:00	-1.44	
16	Ballyglass	01/08/2009 10:36:00	-1.396	

Figure 22 Irish Tides Microsoft Excel Report

OSS users now have the ability to generate reports in word and excel, which did not previously have. Once the report has been published to word users can then customize the report by adding further content to the document or change the styling of the headers or table, the excel doc is saved as a CSV file which is a common storage format for the exchange of structured data. The ability for OSS users to generate both of these reports offers them significant improvements over the previously available HTML tables. It's also worth noting that the web application will be deployed as a intranet application within the MI's LAN and that the word and excel reports can only be generated by authorised user's who would also have both word and excel installed on their personal workstations.

Tidal Graphs

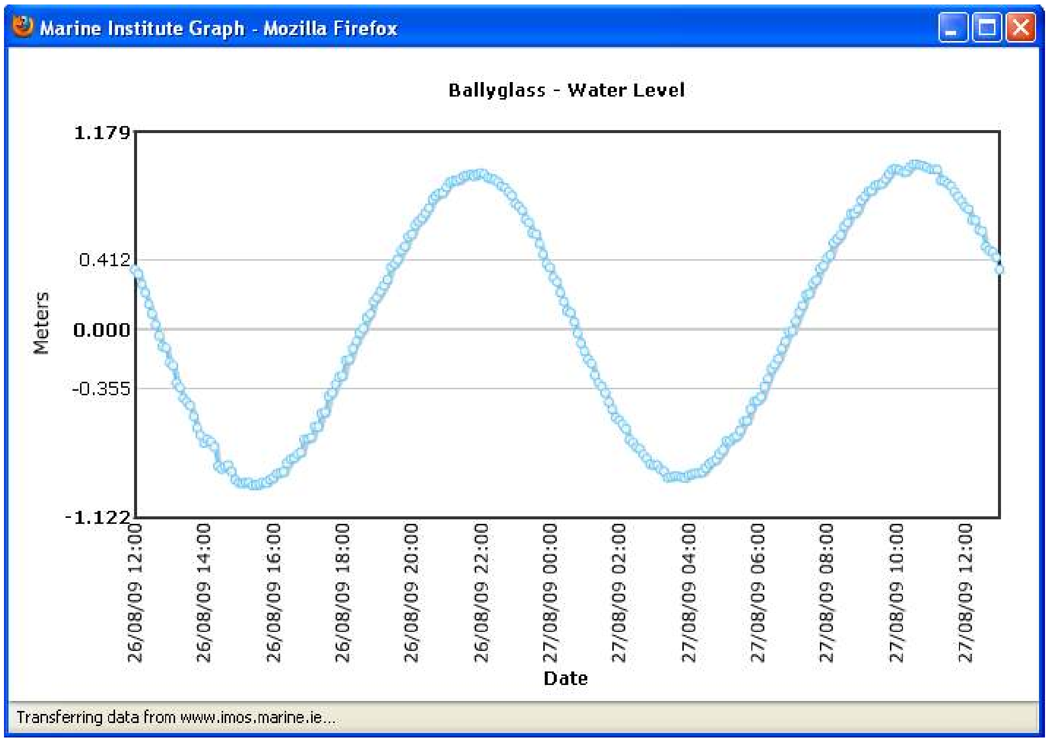


Figure 23 Flash based Tidal Graphing

Previously the MI used two different graphing components in which to render graphs, the first was using crystal reports and another using a flash based solution, both of which required licenses.

Some users using older model of PDA's and cellular phones have been unable to view the Flash based graphs as flash plugins were not supported on these devices. The Flash based component (as shown in *Figure 23 Flash based Tidal Graphing*) also had some limitations with regard to the display format of labels on the X &Y axis. The MI had a requirement to replace the existing graphing components with a new component that does not require a license fee and which can render a graph as an image such as JPEG or a GIF to make it more portable with older mobile devices. They would also like to be able to specify the width and height of the rendered graph, and choose from one of three predefined graph styles.

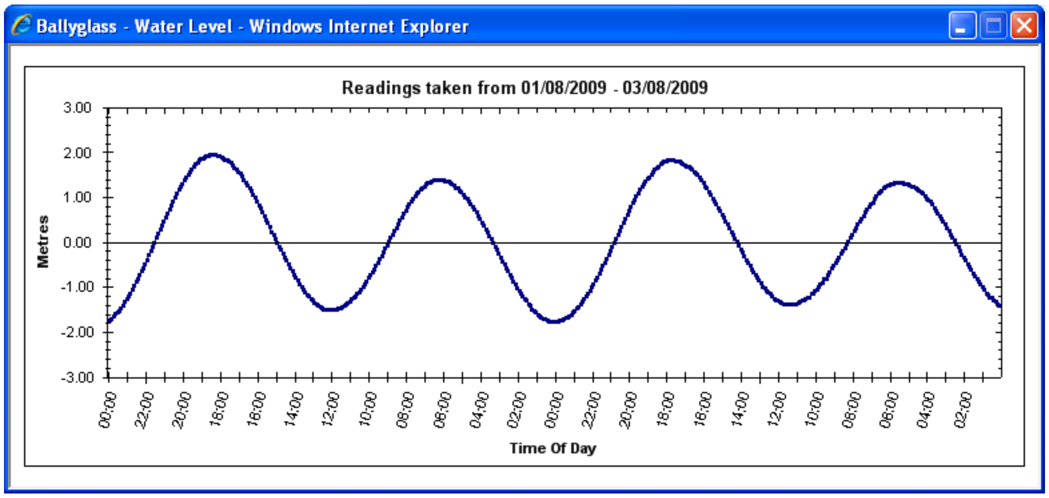


Figure 24 New Irish Tides Graphing - Style 1

After initially trying four different graphing libraries for ASP.NET I eventually settled upon ZedGraph, an Open Source graphing library with offers a high degree of flexibility and custom ability that best suited the MI's requirements. Using the same user-friendly interface as the data reports, with the addition of selectable width, height and graph style options, users can generate custom tidal graphs based on a selected date range and tide gauge location, two of these graphing styles are shown in *Figure 24 New Irish Tides Graphing - Style 1* & *Figure 25 New Irish Tides Graphing - Style 3*.

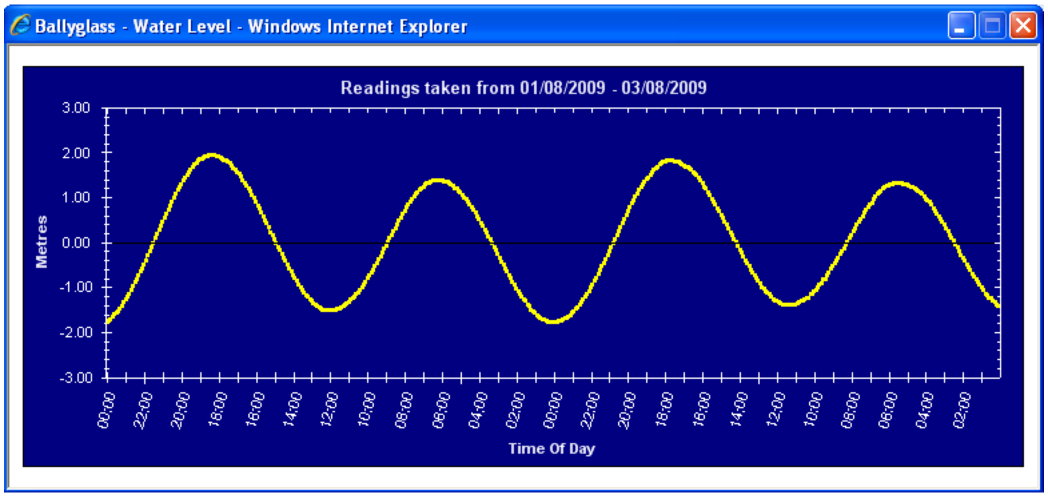


Figure 25 New Irish Tides Graphing - Style 3

The new tidal graphing component is more flexible and customisable for tidal graphing which is a significant improvement over what had previously been offered, as graphs can be generated to an appropriate size selected by the user, and in one of three different styles, and new styles can also easily be added. The ZedGraph component is also available free of charge and does not require any third party executables to be installed on the web server either, you just have to include a ZedGraph library as a reference on your ASP.NET web project. The web page responsible for generating the graph's can also be accessed from outside the web application and does not require users to have been authorised first.

Chapter 7 – Project History

How the project began

I previously worked as a contract software developer for the MI. One of the projects I had worked on was to extract the data from the tide gauges and have it recorded in a database. The Ocean Science Services department had expressed an interest to improve the analysis, reporting and modification of the tidal data but budget constraints had lead to delay in implementing such features. My knowledge of the existing tidal infrastructure within the MI and my participation in the MScSIS made this project an ideal for both me, and the MI.

I started out by having some face-to-face meetings with Dr. Guy Westbrook of the Ocean Science Services department of the MI to obtain some initial requirements for the project.

Project Management

As this project was undertaking as part of my Masters I was the sole developer, researcher, and tester for it's entire duration. All work was carried out on a part-time basis outside of my work hours.

I kept contact with Dr. Westbrook through email and occasionally face-to-face meetings at the MI Head Office in Oranmore, Galway. Dr. Westbrook would provide feedback and answer any questions I may have had. As the project was undertaken using the SCRUM iterative incremental framework I was able to keep Dr. Westbrook informed on a monthly basis with progress reports on the tasks that had been completed during each sprint.

Completed Sprints

Using SCRUM sprints, the project was completed using the following short development cycles.

Table 2 - Completed Sprints

Sprint #	Tasks	Date Delivered
1	• Research suitable architecture framework to be used • Research ASP.NET, C#, and SQL Server 2005 technologies • Research graphing libraries available for ASP.NET • Test a small selection of graphing libraries for suitability to the project	31st January
2	• Set-up database • Create development environment using N-tier architecture	28th February
3	• Create BIZ layer • Create DAL layer • Add List functionality to BIZ layer	31st March
4	• Develop analysis engine for detecting time gaps • Add functionality to analysis engine to send email reports • Implement logging functionality into analysis engine • Update BIZ & DAL layers • Test Functionality	30th April
5	• Develop web form to authenticate users • Develop web form to allow users to specify and select report publishing features • Develop web form to generate Microsoft Work Reports	31st May

	• Develop web form to generate Microsoft Excel Reports • Update Web, BIZ & DAL layers • Test Functionality	
6	• Update custom reports web form to cater for graph reports • Develop graph generator service • Update Web, BIZ & DAL layers • Create/Update Stored Procedures • Test Functionality	30th June
7	• Develop web form to Add Observations • Develop web form to Edit/Delete Observations • Update Web, BIZ & DAL layers • Create/Update Stored Procedures • Test Functionality	31st July
8	• Write up Thesis	28th August

Nearly all estimates were met on time and tasks could be completed in the planned sprints. No significant customer changes were introduced once the project had begun so no new unplanned items had to be added to the project backlog.

Chapter 8 – Conclusion

Lessons Learned

Many lessons were learned throughout the different stages of this project.

It was my first exposure to agile methods and the SCRUM framework, and so I found the agile way of thinking to be more challenging in the beginning. By agile way of thinking I mean developing and releasing in short iterations. Project management in general was totally new to me, and I quickly discovered that I needed be more controlled and always on top of things to meet the sprint goals.

It also gave me a chance to update my skills with technologies I don't get to use every day on the job such as web development with ASP.NET and writing stored procedures for databases. There was also a learning curve with LINQ to entities for mapping the database tables and also the ZedGraph API.

What could be done differently?

It's easy to look back now and say what I would have done differently. I think I would have liked to get the project of the ground earlier and not have had such a rush on to complete development and do a write up in my last few weeks. I would have also liked to develop the web application using the ASP.NET MVC framework, but it didn't go to full release until March 2009 which was much to late to begin development, and with a lack of material available for it also I made a decision to use an N-tier architecture instead.

Technologies Used

The prominent technologies used during this project were ASP.NET, C# and SQL Server 2005. ASP.NET is a web development framework from Microsoft that allows you to build dynamic web applications. C# is used for the ASP.NET code behind pages, for the lower BIZ and DAL layers and also for the windows service that drives the analysis engine.

Meet Expectations

The MI and in particular the OSS dept now have a functional web application in which they can generate both tidal data reports and tidal graphs based on the Irish tide gauge observations, and can also make changes if necessary to any of the tidal observations using web forms developed as part of this project. The analysis service will give daily reports via email to OSS members on any time gaps detected in the tidal dataset. I had three overall goals that had to be met by *"implementing functionality to identify and fix time gaps, and generate custom reports and graphs"* and they were that by doing this I would have improved the analysis, improved the modification, and improved the reporting of the tidal data in Ireland. In chapter 5, I demonstrated how all three of these goals have now been met and how this new system offers significant improvements over the system that previously existed.

Future Plans

Work on this project will continue for the next two months. More work is required on the web form validation, and to add some additional tidal graphing styles. Also at present the email addresses for the analysis reports are stored in the app.config for the windows services, the plan is to make this more dynamic by allowing users to enter these through some web form.

Conclusion and Recommendations

This project has been a great learning experience for me, affording me the opportunity to manage a software development project right from the initial requirements gathering through to design and development. It was interesting to work closely with the customer at all stages, and to have them so engaged in the outcome of the project. I feel I have learnt a huge amount about scheduling and project management while undertaking this masters project. I have also been able to gain some valuable web development experience with ASP.NET and SQL Server.

The new web application and analysis engine also provides the Marine Institute with improved modification, analysis and reporting functionality for the Irish Tides Gauge Network project.

Reference List

Agile Manifesto Organisation. (2001). Manifesto for Agile Software Development. Retrieved August 13, 2009, from http://agilemanifesto.org/

Agile Manifesto Organisation. (2001). Principles behind the Agile Manifesto. Retrieved August 13, 2009, from http://agilemanifesto.org/principles.html

Agile Methodology. (2008). Agile Methodology: Understanding Agile Methodology. Retrieved August 23, 2009, from http://agilemethodology.org/

Control Chaos. What is Scrum?. Retrieved August 13, 2009, from http://www.controlchaos.com/about/

Delaney, K. (2007). Inside Microsoft SQL Server 2005: The Storage Engine (Chapter 2 - SQL Server 2005 Architecture). Retrieved August 14, 2009, from http://library.books24x7.com

Ferracchiati, F., A. (2008). LINQ for Visual C# 2008. Berkeley: Apress.

Machanic, M. (2006). SQL Server DBMS' growing popularity is no coincidence. Retrieved August 12, 2009, from http://searchsqlserver.techtarget.com/news/column/0,294698,sid87_gci1190809,00.html

Marine Institute. Irish Tides Observations. Retrieved August 21, 2009, from http://www.marine.ie/home/publicationsdata/data/IMOS/IMOSITObservations.htm

Marine Institute. About the Marine Institute. Retrieved May 01, 2009, from http://www.marine.ie/home/aboutus/

Microsoft. (2009). .NET Framework Overview. Retrieved August 13, 2009, from http://www.microsoft.com/net/overview.aspx

MSDN. Introduction to Windows Service Applications. Retrieved August 14, 2009, from http://msdn.microsoft.com/en-us/library/d56de412%28VS.80%29.aspx

MSDN. LINQ. Retrieved July 21, 2009, from http://msdn.microsoft.com/en-us/data/cc299380.aspx

MSDN. LINQ to Entities. Retrieved July 21, 2009, from http://msdn.microsoft.com/en-us/library/bb386964.aspx

Schwaber, K., & Sutherland, J. (2007). The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process. Retrieved August 13, 2009, from http://jeffsutherland.com/scrum/ScrumPapers.pdf

Sheriff, P., D. (2002). .NET Development (General) Technical Articles: Building an N-Tier Application in .NET. Retrieved August 12, 2009, from http://msdn.microsoft.com/en-us/library/ms973279.aspx

Softhouse Consulting. Scrum in five minutes. Retrieved August 13, 2009, from http://www.softhouse.se/Uploades/Scrum_eng_webb.pdf

Varallo, V. (2009). ASP.NET 3.5 Enterprise Application Development with Visual Studio 2008: Problem Design Solution. Indiana: Wiley Publishing.

ZedGraphWiki (2007). Main Page. Retrieved July 01, 2009, from http://zedgraph.org/wiki/index.php?title=Main_Page

Annotated Bibliography

Ahari, N., Bartoll, M., & Moldez, O. C. (2004) Design Patterns in C#. Retrieved February 02, 2009, from http://www.mathiasbartoll.com/downloads/programming/Design_Patterns_in_CSharp.pdf

This paper provides a brief introduction to the .NET Framework, the C# Programming Language, and also demonstrates how a selection of design patterns could be implemented in C#. The Design Patterns chosen are Singleton, Abstract, Adapter, Composite, Observer and Strategy. As described by the authors Design Patterns help to provide the foundation for a flexible architecture, which as key to all good object-oriented design.

Marine Institute. Irish National Tide Gauge Network. Retrieved February 03, 2009, from http://www.marine.ie/home/services/operational/oceanography/TideGauge.htm

The Irish Tide Gauge Network is on-going development between the Marine Institute and a number of both public and private organizations, with the aim to develop a permanent tidal monitoring infrastructure that will ultimately consist off between 35 and 40 stations, with a completion date of 2013. This web page is a useful resource for view the latest tide gauge data and at time of writing contains up to date information on 15 tide gauge stations. My thesis project is based solely on the historical data collected from each of these stations.

Murphy, J., O'Mahony, C., Sutton, G., & Woodworth, P. (2003). Scoping Study to assess the status of Irelands tide gauge infrastructure and outline current and future requirements. Retrieved November 14, 2008, from

http://www.agriculture.gov.ie/Fisheries/Engineering/publications/tidegaugereport May2004.pdf

This report contains the recommendations for a new national network of high quality sea level tide stations. It identifies and describes the status of existing gauges that had been used to measure water levels. It then makes proposals for the formation of the now titled Irish Tide Gauge Network, by standardizing instrumentation and technologies used on tide gauge stations, identifying suitable locations for stations, and the cost of implementing and maintaining such a network which would comply with GLOSS (Global Sea Level Observing System) standards. Potential stakeholders were also surveyed in order to ascertain user requirements for the report.

McLaughlin, B. D., Pollice, G., & West, D. (2006). Head First Object-Oriented Analysis & Design. CA: O'Reilly.

This book provides a great introduction to the world of Object-oriented Analysis & Design in a fun and intuitive way through the use of pictures, handwritten notes and use case diagrams. It shows you step by step how to write software that is easy to reuse, maintain, and extend. It brings you right through the software development life cycle from gathering requirements from customers and reacting to changing requirements, to architecturing your software, coding and design principles, and finally testing.

Coronel, C., & Rob, P. (2007). Database Systems: Design, Implementation, and Management (7th ed.). Boston, Massachusetts: Thomson Course Technology.

This seventh edition from Coronel & Rob presents a comprehensive and practical study of database design, development, implementation and management. Database design covers the development of relational database systems, covering normalization, SQL, database performance tuning concepts, query optimization, database connectivity and web development. Implementation deals with how databases are implemented and issues that may arise, while management is concerned with the management issues like transaction management, concurrency control.

Davidson, Louis., Dewson, Robin., Machanic, Adam., Narkiewicz, Jan., Rizzo, Thomas., Sack, Joseph., et al. (2005). Pro SQL Server 2005. California: Apress.

The chapters in this book are divided between the eight authors who each provide some practical and in-depth coverage of the topics they each specialize in. The book takes a closer look at the new range of functionality and features provided by SQL 2005 such as T-SQL programming, .NET integration, the XML Datatype, and Reporting Services. It also includes sections covering more DBA oriented tasks like the Installation and Configuration of SQL 2005, Maintenance, Database Mirroring & Security, Notification Services and also some of the new Management tools.