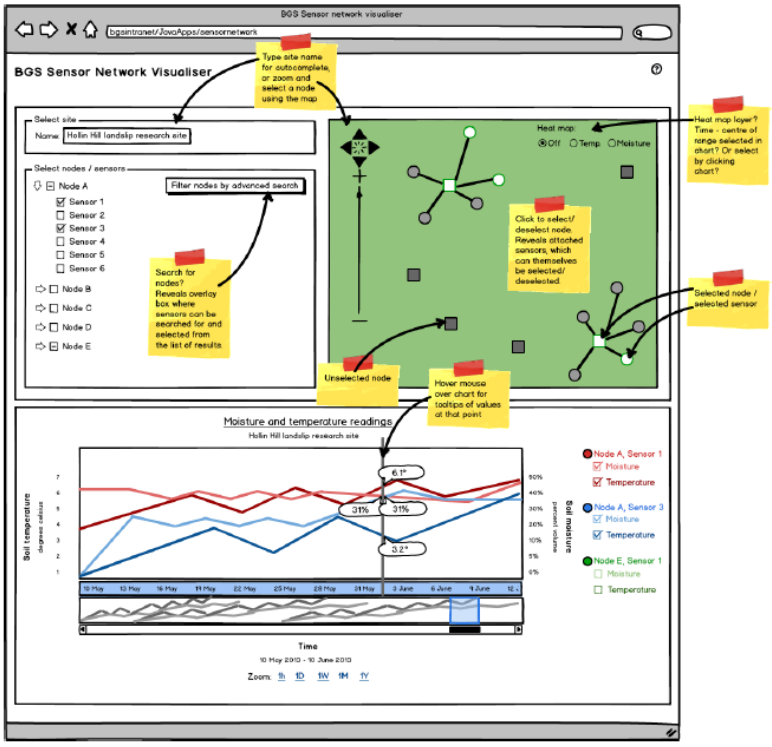
British Geological Survey
NATURAL ENVIRONMENT RESEARCH COUNCIL

SensorNet API Development Report

Informatics Programme

Internal Report IR/14/037



BRITISH GEOLOGICAL SURVEY

INFORMATICS PROGRAMME

INTERNAL REPORT IR/14/037

SensorNet API Development Report

Carl Watson, Graham Smith, Nick Russell

Contributor/editor

Philip Meldrum, Garry Baker & Martin Nayembil.

Keywords

Sensor; Web portal; Development; API .

Front cover

Cover picture produced by Nick Russell using the Balsamiq software: http://balsamiq.com/

Keyworth, Nottingham British Geological Survey 2012

BRITISH GEOLOGICAL SURVEY

The full range of our publications is available from BGS shops at Nottingham, Edinburgh, London and Cardiff (Welsh publications only) see contact details below or shop online at www.geologyshop.com

The London Information Office also maintains a reference collection of BGS publications, including maps, for consultation.

We publish an annual catalogue of our maps and other publications; this catalogue is available online or from any of the BGS shops.

The British Geological Survey carries out the geological survey of Great Britain and Northern Ireland (the latter as an agency service for the government of Northern Ireland), and of the surrounding continental shelf, as well as basic research projects. It also undertakes programmes of technical aid in geology in developing countries.

The British Geological Survey is a component body of the Natural Environment Research Council.

British Geological Survey offices

BGS Central Enquiries Desk

Tel 0115 936 3143 Fax 0115 936 3276

email enquiries@bgs.ac.uk

Environmental Science Centre, Keyworth, Nottingham NG12 5GG

Tel 0115 936 3241 Fax 0115 936 3488

email sales@bgs.ac.uk

Murchison House, West Mains Road, Edinburgh EH9 3LA

Tel 0131 667 1000 Fax 0131 668 2683

email scotsales@bgs.ac.uk

Natural History Museum, Cromwell Road, London SW7 5BD

Tel 020 7589 4090 Fax 020 7584 8270

Tel 020 7942 5344/45 email bgslondon@bgs.ac.uk

Columbus House, Greenmeadow Springs, Tongwynlais, Cardiff CF15 7NE

Tel 029 2052 1962 Fax 029 2052 1963

Maclean Building, Crowmarsh Gifford, Wallingford OX10 8BB

Tel 01491 838800 Fax 01491 692345

Geological Survey of Northern Ireland, Colby House, Stranmillis Court, Belfast BT9 5BF

Tel 028 9038 8462 Fax 028 9038 8461

www.bgs.ac.uk/gsni/

Parent Body

Natural Environment Research Council, Polaris House, North Star Avenue, Swindon SN2 1EU

Tel 01793 411500 Fax 01793 411501

www.nerc.ac.uk

Website www.bgs.ac.uk

Shop online at www.geologyshop.com

Acknowledgements

We would particularly like to thank Martin Nayembil and Phil Meldrum for the initial vision and database structures which power the web based SensorNet prototype.

Contents

Summary

This report describes the development of a prototype web based application for the dissemination and analysis of sensor network data as held in the corporate database for virtual sensor data, referred to throughout this report as SensorNet.

The first part of the report introduces the project and the context of this particular bit of work.

The following sections describe the approach taken, the technologies considered, how long each element of the work took, a summary of the results so far and an outline of future tasks.

1 Introduction

Data has been collected from a Wireless sensor network (WSN) at the Hollin Hill Observatory site, located North East of York, since November 2011. The site was established as part of a GTom and Soils program initiative between Phil Meldrum and Barry Rawlins to investigate how variation in soil moisture is affected by factors such as soil texture and topography that operate at different spatial scales. It has been used to demonstrate Sensor-Net as example of an environmental observatory where data can be captured, transferred and analysed close to real-time. Eight Nodes are distributed across the site with twelve sensors connected to each node. Each sensor delivers measurement data for Moisture, Temperature and Electrical conductivity. Measurements are scheduled to be collected every fifteen minutes delivering a total of over eight-hundred-thousand measurements a month. Data is stored locally at the Sensor Node and then transmitted across a radio link to a central station at site which uses a GSM link to re-transmit this data onto a central server. To ensure that these measurements were dealt with effectively, Martin Nayembil started work on building an open architecture database scheme to store the sensor data and make it readily available to scientists. The need to easily access and visualise this sensor data from anywhere was the impetus to build a user friendly interface to the SensorNet data

This report does not describe the ingestion process or the design of the database for holding sensor (or virtual sensor) data, for more details on these elements contact Martin Nayembil.

2 Method

This section provides a summary of the main actions and approach taken to produce the proof of concept prototype, where appropriate links relevant resources are provided.

2.1 APPROACH TAKEN

A small development team of Carl Watson, Graham Smith and Nick Russell were tasked with creating the prototype system by the end of the 2013/14 financial year. Carl would oversee initial requirements analysis and high level project management duties with Graham and Nick producing the server side and client side code respectively.

The initial actions carried out ensured that the development team understood the requirements before too much time was spent on research and coding. A series of conversations were held to refine the requirements, rank priorities and agree timeframes.

Once the development team had a high level understanding of what was required a solution design session was held, it was agreed that a general web based client server architecture would be used that incorporated existing BGS server components that provided corporate database content to web client that was optimised for a rich user experience. A number of external graphing libraries, APIs and standards were discussed as candidates for inclusion in the client and Nick agreed to carry out a desk study of the visualisation toolkits Rickshaw (http://code.shutterstock.com/rickshaw/), dygraphs (http://dygraphs.com/) and D3js (http://d3js.org/).

As someone involved in the development of the PropBase architecture Graham Smith was keen to re-apply that architecture to the SensorNet problem, this would leverage existing internal code whilst expanding the scope for which it was initially developed and provide an additional example for possible new query layer applications.

The development team adopted an iterative approach to this work in a number of ways, firstly the customer (Phil Meldrum) would be consulted on a regular basis to discuss progress, address any issues and agree next steps. The development of client side code had a direct impact on the design of the server side interfaces and vice versa, therefore regular contact was needed between Graham and Nick to ensure the two sets of code complemented each other as the prototype evolved.

As part of the design phase, a simple user interface mockup was produced using the software balsamiq (http://balsamiq.com/) and presented to the customer to ensure that the development team were on the right track, see Figure 1.

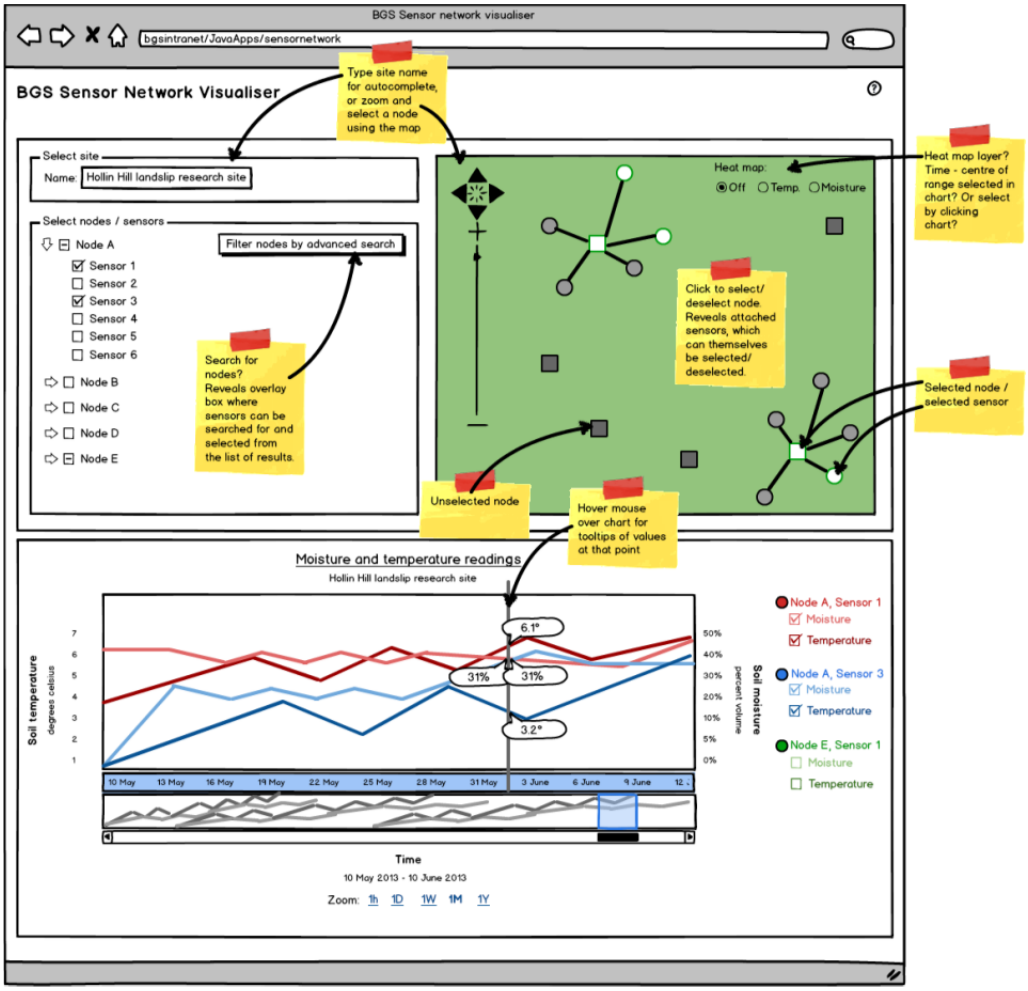


Figure 1: Balsamiq mockup

Once the design and architecture were agreed a period of research and development took place in which the selected visualisation toolkits were implemented, assessed and ultimately demonstrated in a stakeholder meeting, January 2014. By the end of the stakeholder meeting it was agreed that the prototype should be completed using the D3js technology, this decision was based on performance and functionality as well as the recommendations of Nick Russell, who although new to JavaScript had now acquired a significant amount of experience through research and testing.

3 Results

This section describes the prototype architecture and user interface produced to satisfy all of the high priority requirements as agreed in the January stakeholder meeting. The main aim was to produce a web based interface that displayed line graphs, using D3js, of real sensor data from Hollin Hill.

The prototype should allow a user to:

- Select a date range
- Select a single sensor
- Select up to two properties to display
- Render a line chart based upon the dates/sensor and properties selected
- Zoom/pan chart to a selected time range
- View node and sensor locations on a map
- View user interface on a variety of screen sizes (i.e. UI restyles according to size)

3.1 PROTOTYPE ARCHITECTURE

The prototype was constructed using the same high level architecture as developed for the BGS PropBase system, Figure 2. The design was adapted to meet SensorNet requirements by defining new data sources, in the form of sensors and virtual sensors, a new optimised query layer and sensor specific queries were developed which power the client facing web services. The JavaScript Client selected was a combination of bespoke code developed by Nick Russell and the external toolkit D3js, jQuery, Bootstrap and the GoogleMaps API.

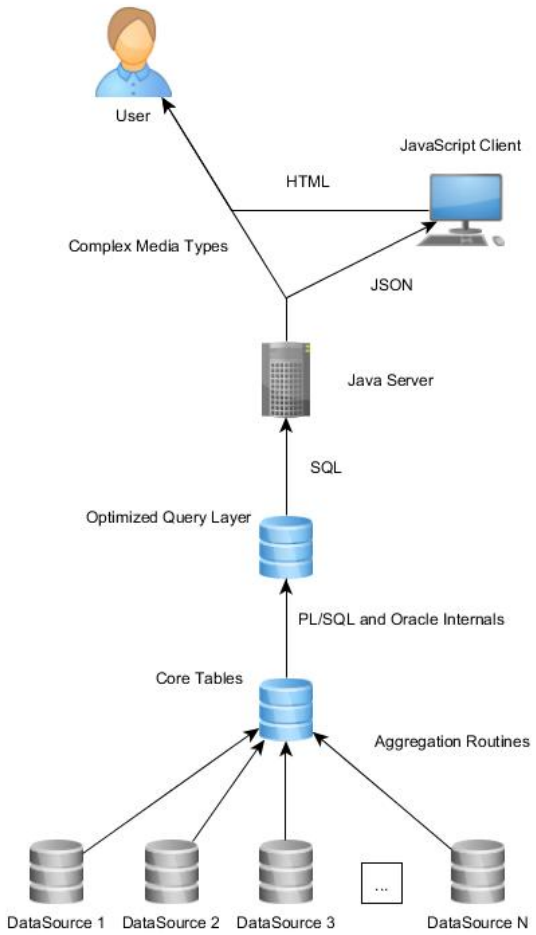


Figure 2: PropBase derived architecture

3.2 PROTOTYPE USER INTERFACE

The sequence of user interfaces shown in Figure 3 represents the key stages of client side prototype development, the screen shots are ordered with the earliest design first, with the end of FY deliverable and the most mature design shown last.

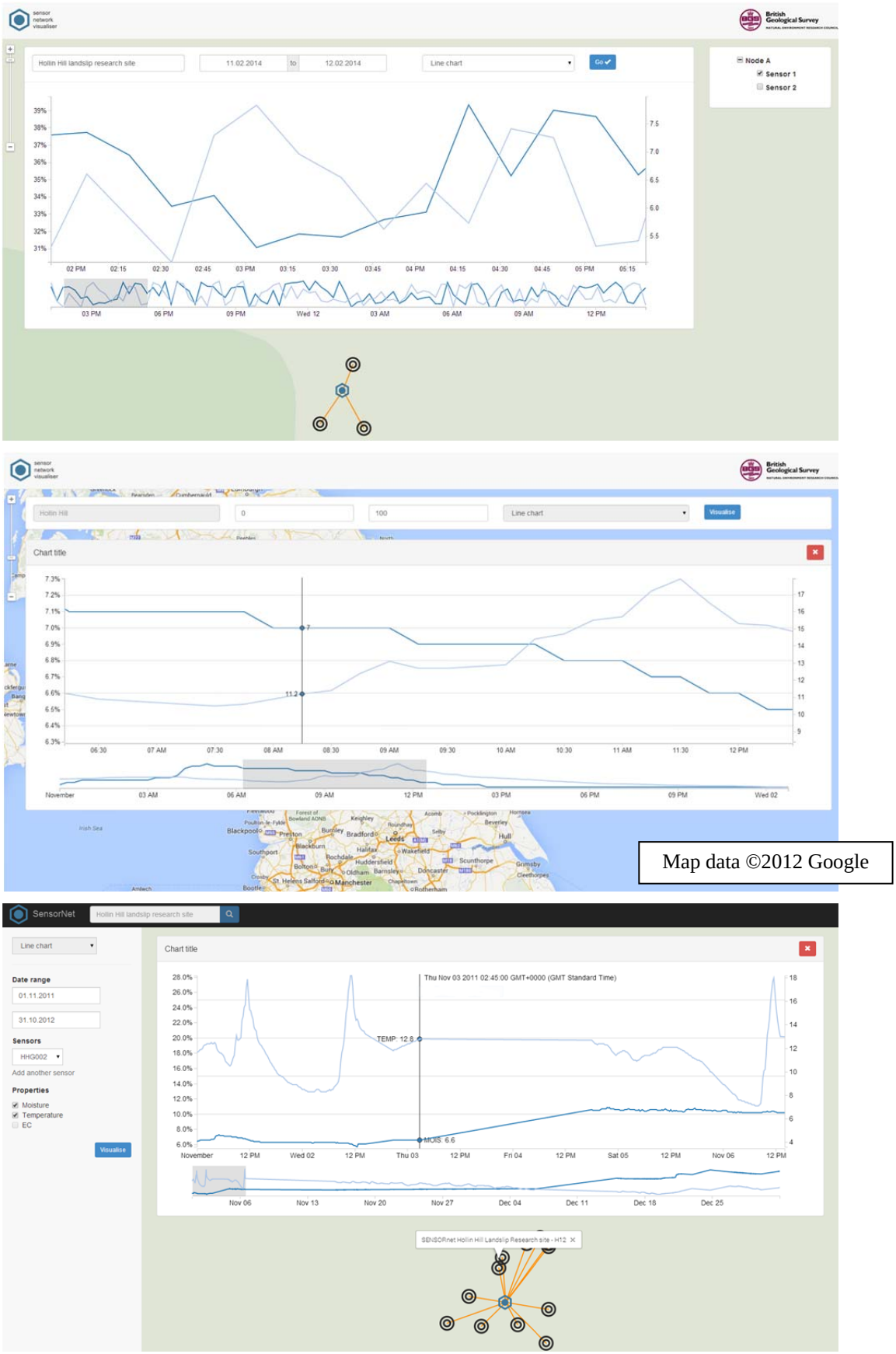


Figure 3: The three stages of User Interface development in order of increasing maturity

4 Conclusions

In order to capture what lessons were learnt during the development of this prototype this section provides a summary of any difficulties or issues encountered and attempts to give guidance on the implications for future developments. Work carried out so far has been logged in the issue tracking software Jira, this provides us with a means of tracking effort and helps us to provide estimates on future tasks. So far, approximately 160 hours have been spent on this work by the three individuals identified as the development team, this does not include the efforts by Martin Nayembil, Phil Meldrum or any other staff involved.

4.1 WHAT WORKED WELL, WHAT TOOK A LONG TIME...

4.1.1 Researching JavaScript and possible library options

A significant block of time was allocated for investigation into the various JavaScript libraries and toolkits that provided graphing functionality, this appears to have been time well spent in that it gave us a greater understanding of the capabilities and limitations of what was available. Some of this time was spent getting familiar with the principles and existing libraries around the JavaScript language which is not a technology heavily used, by this developer. Developing greater JavaScript skills increases our capacity to deliver more dynamic and responsive user interfaces, within web browsers. There remains a lot to learn about this language and its qwerks, much of the learning process will take place as part of the testing and development of JavaScript code.

Approximate time spent: 5 days

Estimate of time for future reviews: 2-? days depending on requirements

4.1.2 Testing JavaScript Libraries and selecting a favourite

When trying to assess the relative merits of the competing options there was a lack of predefined assessment criteria on which to judge performance, this may have delayed how quickly certain options were removed from the process. However, due to the capacity building element of this work it was still time well spent and would help us in the future should we wish to look at alternative JavaScript (or non-JavaScript) libraries.

Approximate time spent: 15 days

Estimate of time for future testing: 5-15days depending on requirements

4.1.3 Client side chart types

Most of the JavaScript libraries considered offer the ability to render temporal data in a range of chart types, the prototype development focussed on the implementation of the line chart. Adding a new chart type to the prototype interface should be a relatively straight forward task so long as that chart type relates to the same sort of temporal data used in the production of the line charts. The selected JavaScript library for charts (D3) is capable of displaying almost any 2D representation of data, the more clearly we can define the chart requirements, the quicker it could be added. The steps involved include; learning the chart basics, implementation using real data, reworking the server link where the data model is not an automatic fit, optimise performance and make the user interface easy to use and intuitive.

Approximate time spent: 10 days

Estimate for addition of 1 new chart type: TBC as it depends on the changes required, minimum of ½ day requirements gathering. Developer time of 5-10 days likely, depending on chart type

4.1.4 Styling user interface

One of the aims of this prototype has been to produce an easy to use, intuitive user interface. This is an activity that has implications for all client side developments, it is difficult to quantify the effort required by it is fair to say that this is an area that the corporate systems development team have not focussed on too much in the past. It involves more time spent on the design of a system and less on the coding, so far this has largely been carried out by Nick Russell without too much input from the users, this is a pragmatic way to create an initial prototype but there should an opportunity to incorporate user feedback at a later date.

Approximate time spent: included within other activities

Estimate for future styling: hugely variable as it depends on how users react to the prototype, we recommend that each user interface development should include no less than 20% contingency for styling.

4.1.5 Heatmaps

The production of heatmaps from sensor data was discuss and deemed to be a nice to have and was not developed for the end of year prototype. It is difficult to estimate how long this would take to produce but it is not anticipated to be overly complex, the BGS have some experience of this in the GIS development team and Andrew Marchant or Wayne Shelley should be consulted if and when this functionality is scheduled for development.

Estimate 10 days

4.1.6 Client side mapping functions

The prototype has evolved through several iterations of a mapping interface, most recently this has involved the use of the GoogleMaps API, the use of this and alternative open source options has been relatively straight forward and the BGS have a wealth of experience in the development of web based GIS style interfaces, using GoogleMaps, OpenLayers and ESRI APIs as appropriate. The functionality in the prototype has been limited to a simple rendering of the sensor locations on a map, these sensor points are located according to coordinates acquired from the database and can be selected by a user through mouse clicks.

Approximate time spent: 5 days

Estimate for future mapping functionality: TBC as it depends on the changes required, minimum of ½ day requirements gathering

4.1.7 **Adding a new sensor property** (including Ancillary sensor data such as battery power)

So long as the form of the new property was similar to the existing properties and were implemented in the database in the same way it would be easy to add these into the client-server code and display them in the user interface. Some additional time would need to be spent ensuring that the interface remained easy to use as more and more properties were added. Some additional work may be required server side as well as client side to present this data.

Estimate 4 days

4.1.8 Server Side coding

The prototype was initially cloned from the Propbase equivalent service and all the generic components retained. However All components specific to Propbase were removed and or repurposed for the differing needs of the sensor dataset, In particular the core datamodel and database access layer, from which it is returned has been reworked from scratch, as needs to occur for any largly different set of data objects. This process took about 5 days to analyse the model and construct basic data retrieval mechanisms. It is envisioned that only minor changes to

this model shold be needed point forward to add additional parameters and additional sites returning the same parameter data should be available without change.

Approximate time spent: 5 days

The database access layer has since been expanded to include a number of different filters as required by the URL designs produced in 4.1.9. This additional set of filters took about a day to add.

Approximate time spent: 2 days

Additional caching mechanisms have been added based upon those in use in the Propbase code, and as these are largly generic objects they were quick to add however it should be noted that beyond this it is possible that additional non-trivial optomisations may be required along side database side optomisations.

Approximate time spent: 0.5 days

4.1.9 URL Design

The URL design provides the Client interactions with the dataset and this is based upon both the data model created in 4.1.8 and the client's interactions designed as part of the charting requirements, this process included some investigation of date formats that had not previously been carried out in the propbase server and took about a day to design and an additional 2 or three days to implement in the server code. A proportion of the code created here is generic and reapplicable to other applications.

Approximate time spent: 2 days

4.1.10 Data Transfer format (JSON)

The current data transfer format in JSON is a generic model of the data and closely follows the format of the internal data model in Java this was relatively quick to complete and provides the currnet code to the clients. Unfortunatly this format being generic is very verbose and could be enhanced and slimed down. To reduce transfer time to the client. A more specialised JSON format (or formats) is recommended once charting requirements become even clearer, these sit close to the lower end of the complexity scale. Additional formats for the data including HTML, KML, and others can also be produced depending upon user and client needs. It is also recommended that the HTML formats be added as these are often useful for developers as a working exampler client, in order to develop additional clients.

Approximate time spent: 1 days

Estimate 1-5 days per fomat depending upon complexity of the data structure required and wether or not we have library code available already.

4.1.11 Security and login functionality

The prototype does not contain any security or login controls, it is the development teams recommendation that ½ a day is spent investigating the requirements, we expect that existing BGS login code could be applied to the prototype, server. It is worth noting that the BGS do not own a valid HTTPS certificate (http://en.wikipedia.org/wiki/HTTP_Secure) which would give both user confidence in the service we provided and encryption between client and server.

Estimate 10 days

4.1.12 Creation of reports and system alerts

The prototype does not contain any reporting or functionality to set up alerts. Setting up a modest number of standard reports, based upon the already displayed sensor data, would be a relatively

simple affair but requires clear definition from the users on what such reports would contain. We recommend that 2 days requirements gathering would clarify what reporting (and alert management) is required.

If the reporting requirements are relatively simple, and do not provide users with a means to alter existing or create new reports, the coding is likely to take a few days. Martin Nayembil (Data Architect) may be required to implement new optimised views that provided decoded query layers.

Estimate 7 days

4.1.13 Adding completely new datasets

There are two likely approaches to adding in new temporal datasets into the system, the simplest from the perspective of the prototype development team would be to ingest the data from the virtual sensor database developed by Martin Nayembil. This would require significant time to be spent on ingesting that data into the standardised virtual sensor data model and Martin should be consulted on a dataset by dataset basis.

Estimate 1-25 days, this wide range is due to the wide range of possible datasets which could be added, the closer it resembles the existing dataset the smaller the time required

Another approach would be to develop a way of incorporating alternative datasets at the client side, for example; if the Met Office provided a rainfall web service that we wanted to display in association with Hollin Hill sensor data we could bypass the database and display the data in the line graph visualisations. This would be completely new functionality and potentially result in a significantly slow system due to the extra work being performed on the client side, but it is possible.

Estimate – This is a significant redesign of the system and would probably take at least 25 days across the development team members, requirements would need to be gathered, solutions discussed amongst the team and with both Martin Nayembil & Phil Meldrum.

Glossary

Prototype The server-client code and user interface developed by the end of the financial year 2013/14 to display sensor data from the Oracle SensorNet database designed by Martin Nayembil.