Newmont 3

Patrick Barringer Tomas Morris Emil Marinov

June 18, 2013

Contents

I Introduction **2**

1 Client 3

2 Product vision 3

II Requirements **3**

3 Overview 3

4 The Model 3

5 SharePoint 3

6 Data Collection UI 4

7 Web Service 4

8 Final Deliverable 4

III System Architecture **4**

9 Client UI 5

10 Web Services 5

11 SharePoint 6

12 Download UI 6

IV Technical Design **6**

13 UI 6
- 13.1 Model Selector . . . 6
- 13.2 File Selector . . . 7
- 13.3 Other Windows . . . 8
  - 13.3.1 Add File Type . . . 8
  - 13.3.2 Metadata . . . 8
  - 13.3.3 Add Metadata . . . 8

14 SharePoint 8

V Design And Implementation Decisions **9**

15 Introduction 9

16 The Client UI 9
16.1 Language choice . . . 9
16.1.1 The UI Itself . . . 9
16.1.2 The Definition Language . . . 10
16.1.3 Metadata . . . 10
16.2 Page Presentation . . . 10
16.3 Windows 8 UI . . . 10
16.4 XML . . . 10
16.4.1 Projects . . . 10
16.4.2 Requirements . . . 11
16.4.3 File Storage . . . 11

17 SharePoint 12
17.1 Framework Choice . . . 12
17.2 SharePoint Settings . . . 12
17.3 Web Part . . . 12

VI Results **12**

18 UI 12

19 SharePoint Web Part 13
19.1 Deployment . . . 13

List of Figures

1 The System architecture . . . 5
2 The Project Selection Interface . . . 6
3 The File Selection Interface . . . 7
4 Other windows . . . 7
5 The SharePoint List . . . 8
6 The Project Description On SharePoint . . . 9

Part I
Introduction

1 Client

Newmont Mining Corporation is one of the world's largest producers of gold, with active mines in Nevada, New Zealand, Australia, Indonesia, Ghana and Peru.

2 Product vision

Newmont has digital records of information critical to the performance of their mines. These records change often and Newmont needs a way of maintaining a centralized repository of these records. The goal of this project was to build said centralized repository.

Part II
Requirements

3 Overview

Newmont maintains collections of related files (a model of an orebody, a "model") that change often, and they need a way of maintaining their collection of these models. They need a method of retrieving a model remotely. To do this we needed to create 3 things: a SharePoint site, a data collection UI, and a web service.

4 The Model

A model consists of a collection of files that describe some aspect of the mine (e.g. topography, gold grade, copper grade, etc.). These documents can also have accompanying documentation (in the form of Excel or Word documents). These files may be scattered across multiple directories and mapped drives. There are files that must be included in a model (e.g. topography), there are others that are optional, and there are some that are unique to one model. Since these files change there is also associated metadata, namely a timestamp indicating when it was last modified and by whom.

5 SharePoint

The SharePoint site will be the reference library for the models (it can not be used to store the models directly because of size constraints). SharePoint will also be the source of the

configuration files (what files must be included in models and the like), as well as (at the very least) the initiating step for a download of a model.

6 Data Collection UI

This will be the piece of software that is deployed to the field. It will allow the user to select files to include in the model (it will have to convert the paths to these files to their original UNC paths to avoid mapping problems). This software will also initiate the upload to the SharePoint site.

7 Web Service

This will be the glue that holds the entire application together. It will handle the transfer of the large files as well as the communication between the data collection UI and the SharePoint site.

8 Final Deliverable

Newmont does not need this to be integrated into their existing code base due to the security requirements of their intranet. They are looking for a proof of concept that they can build on and then integrate.

Part III
System Architecture

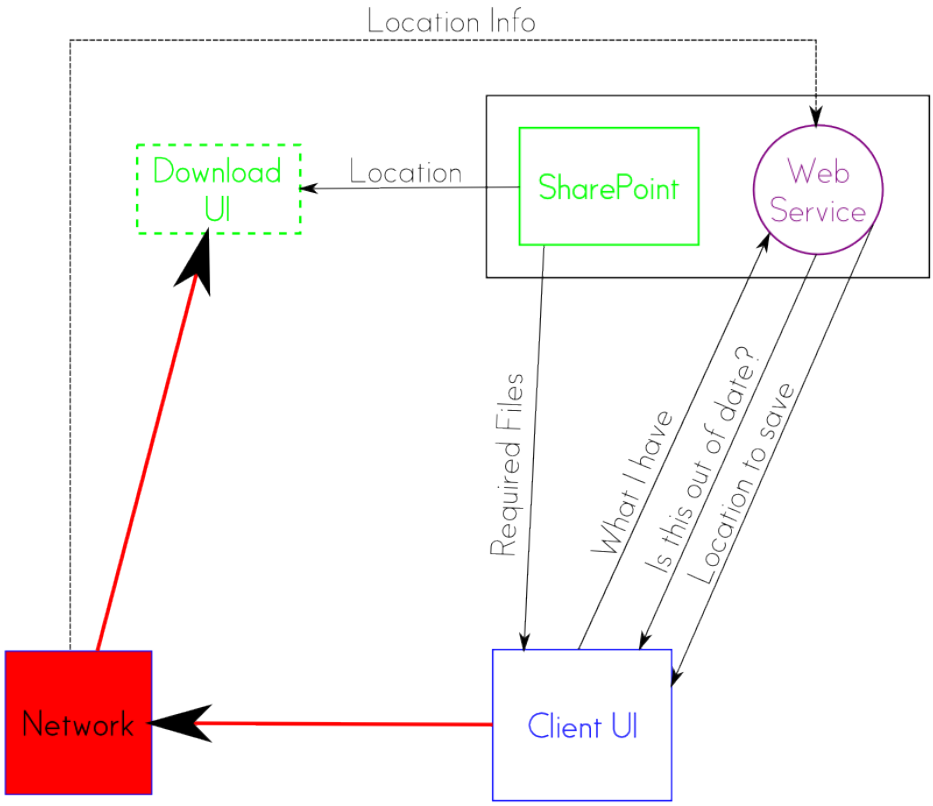

Figure 1: A graphical representation of the system architecture

9 Client UI

The Client UI is a WPF application that connects to the SharePoint through an http connection. It takes in data about the models (location, files, metadata) and shows them to the end user.

10 Web Services

The web services in this case are a feature of SharePoint and are provided by the SharePoint Client Object Model for C#. These include being able to download, upload, and update files from the SharePoint and also being able to pull version history from the SharePoint.

11 SharePoint

The SharePoint represents the administrative front-end for the workings of the application. It allows access to the XML that dynamically generates the client UI and also the download of the zip file created by the client UI.

12 Download UI

The Download UI was subsumed by the SharePoint site because it could handle that functionality.

Part IV
Technical Design

13 UI

13.1 Model Selector

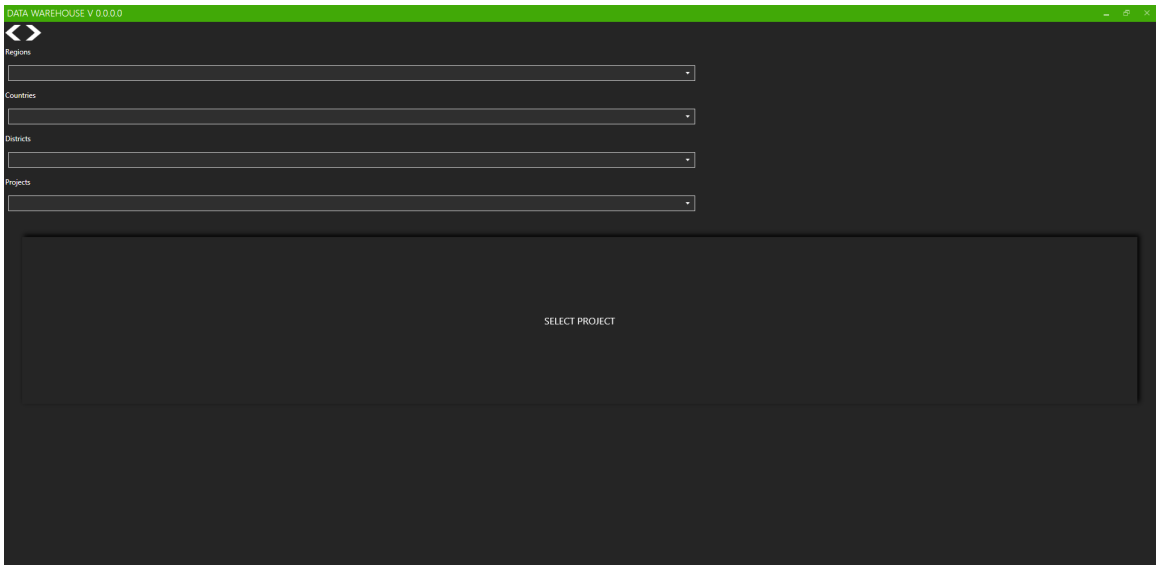

Figure 2: The Project Selection Interface

Figure 2 shows the interface that allows the user to select a model, its fields are populated from the SharePoint. When the user selects a model s/he is taken to the file selection page.

13.2 File Selector

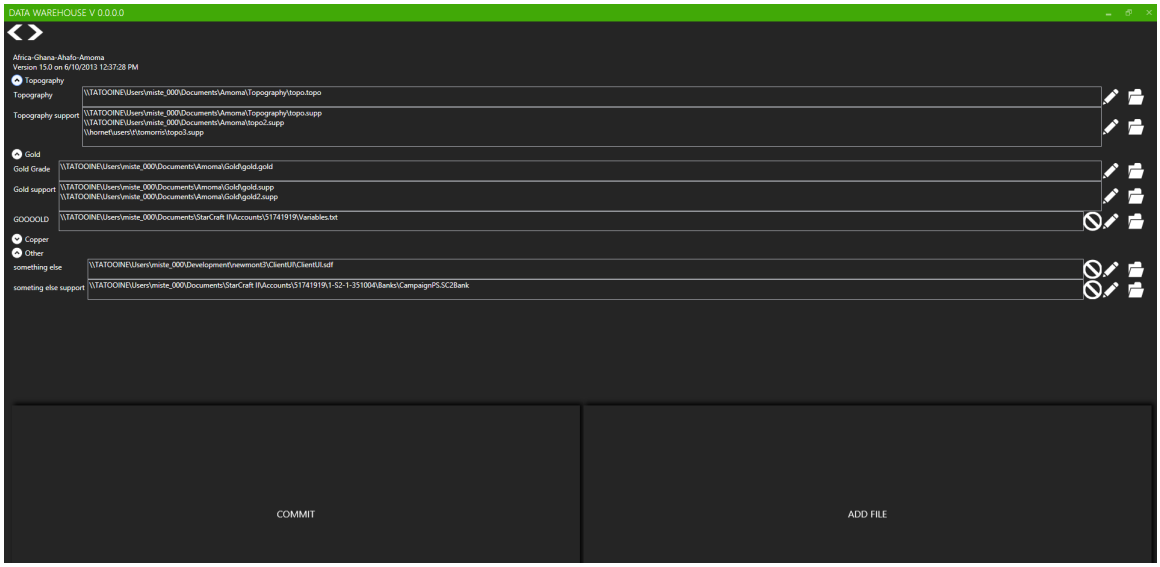

Figure 3: The File Selection Interface

Figure 3 shows the interface that allows the user to select files to include in the model (selected using the folder button next to a row, or by typing the file path into the text box [not recommended]), it also provides the option to add new files (via the add file button), remove optional ones (via the crossed out circle), and edit the metadata associated with them (via the pencil button).

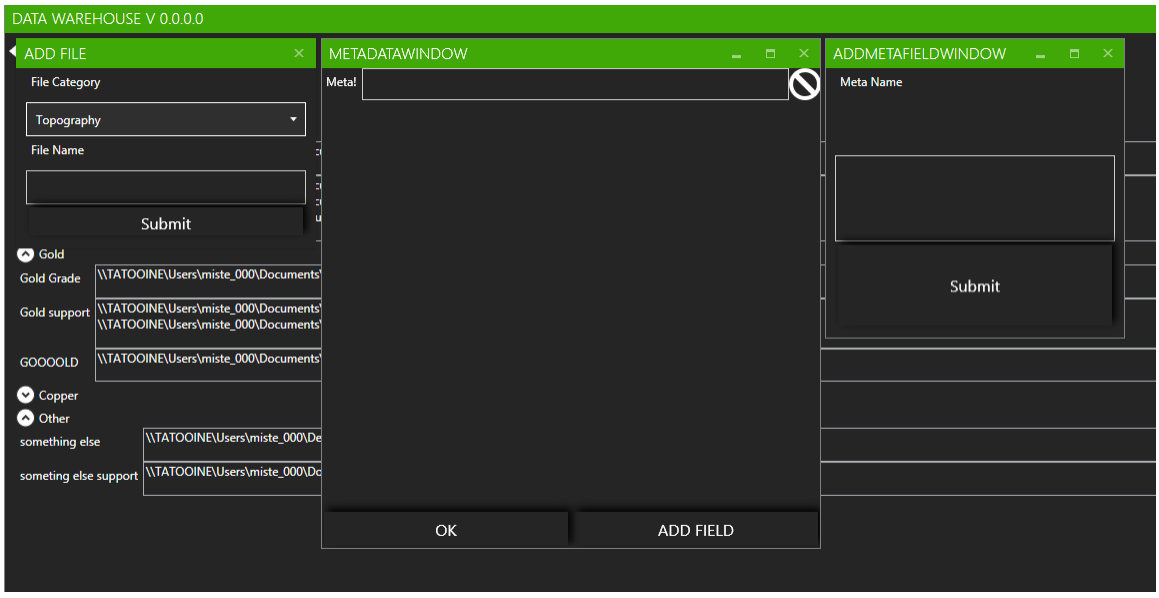

Figure 4: from left to right: the window for adding a file, the window for modifying metadata, and the window for adding a metadata field.

13.3 Other Windows

Figure 4 shows the windows used to provide additional functionality.

13.3.1 Add File Type

The first window from the left shown in figure 4 allows the user to add a new type of file to the file selection page. This file type will be added to the XML document that is generated on commit. When this model is selected in this interface after the upload this file type will still be present. (It will not be present on any other models however.). The add file dialog box requires the user to select a category for the new file type, this is selected through a dropdown menu.

13.3.2 Metadata

The middle window in figure 4 shows the dialog for editing the metadata associated with a file. If there are options provided (e.g. metric or imperial for units of measure) there is a drop down menu presented, otherwise there is a free entry textbox.

13.3.3 Add Metadata

The rightmost window in figure 4 is the window for adding an new metadata "trait" it is almost the same as the add file type window, the only major difference is there is no catagorical ordering of metadata. (Again on commit any new metadata is persisted to the xml so it will show up in later revisions of this model)

14 SharePoint

Figure 5: The SharePoint List

The tree in figure 5 uses the same xml metadata file from the model selector and populates a tree that represents all of the models that could have information stored on the SharePoint. Clicking any node loads additional data on that node.

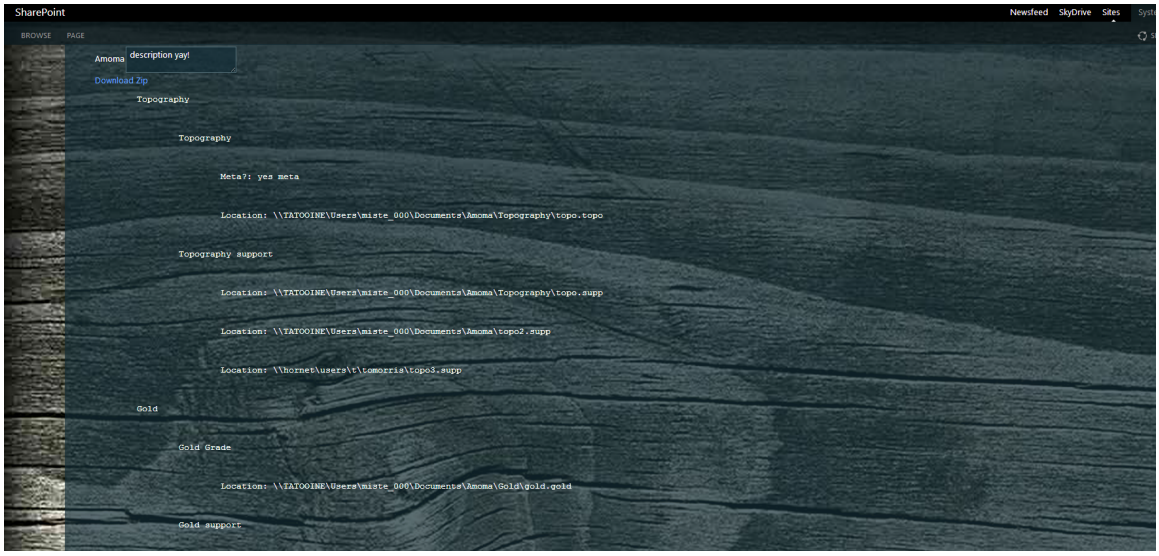

Figure 6: The Project Description On SharePoint

The area in figure 6 is populated on click of a node from the tree in Figure 5 and loads any metadata that is stored on the SharePoint. If the node is a model, it enables a link to download the latest zip of the model files.

Part V
Design And Implementation Decisions

15 Introduction

As was stated in part II we are building a data warehouse for Newmont's models of various orebodies. This consists of two different parts: a client UI and a web storage site.

16 The Client UI

16.1 Language choice

16.1.1 The UI Itself

We used the WPF suite of Microsoft .NET to build the UI because that is what our client asked us to use and it effectively filled the requirements of the client.

16.1.2 The Definition Language

The clients wanted the UI to be populated from a file that can be changed by management with ease. We decided to use XML to define the items that would be added to the UI. This was done because XML is a human readable language. It is easy to grasp the structure of XML even without a programming background, and it can be easily parsed by C#.

16.1.3 Metadata

To store the metadata that accompanies each model file we also chose to use XML. We did this for the reasons listed in the previous section and to provide consistency to the end user that would configure this client application.

16.2 Page Presentation

We put the "choose your model" portion of the UI on a separate page from the "model building" section because this is what the client asked for, and because it made the most sense. This allows for the "model building" section to be populated from the web with relative ease. The final part of the UI, the metadata for each file, is presented in a new popup window. This was done because it prevented the model building section from becoming to cluttered.

16.3 Windows 8 UI

The switch to Modern UI (or the Windows 8 style) was performed for purely aesthetic reasons based on the personal tastes of a member of the design team. This has no impact on the portability of the application, it is usable on any PC with Windows.

16.4 XML

16.4.1 Projects

All of the data on the models and the files associated to them is stored on the SharePoint site in the form of XML documents. This allows easy data manipulation and integration into the UI. This also allows easy manipulation of the client UI without knowledge of programming or the code base.

```
1 <Root>
2    <Region description="description yay!" name="Region1">
3      <Country description="description yay!" name="Country1">
4        <District description="description yay!" name="District12">
5            <Project description="description yay!" name="Project1" type="In-
     Situ"/>
6        </District>
7      </Country>
8    </Region>
```

```
</Root>
```

Explanation This matches the hierarchy of Newmont's models, they are broken down by region then country then district then models which have a type.

16.4.2 Requirements

```
<Files>
  <Category Name="Other">
    <File Name="something else" Optional="True">
      <Meta Name="Unit of Measure">
        <Type Name="MKS" />
        <Type Name="Imperial" />
      </Meta>
      <Meta Name="Something else" />
    </File>
    <File Name="something else supporting docs" Optional="True" Many="True" />
  </Category>
</Files>
```

Explanation This file allows the model building part of the UI to be populated according to the wishes of management. Files are broken down into categories which can have multiple files, these files can then have meta data added to them. If the meta data has types a drop down list is presented to the user, otherwise a free-form text box is displayed.

16.4.3 File Storage

```
<Project name="Project1" fullname="Region1-Country1-District12-Project1">
  <Category name="fileCategory">
      <File name="something else" >
          <Location value="//Patrick-PC/dir/to/project/file" />
          <Meta name="Unit of Measure" Value="MKS" >
              <Type name="MKS" />
              <Type name="Imperial" />
          </Meta>
          <Meta name="Something else" value="some value" />
      </File>
      <File name="something else supporting docs" >
          <Location value="//Patrick-PC/dir/to/project/file/s1" />
          <Location value="//Patrick-PC/dir/to/project/file/s2" />
      </File>
  </Category>
</Project>
```

Explanation This file stores the location(s) and metadata for the uploaded files for each model.

17 SharePoint

17.1 Framework Choice

We chose to use SharePoint server because that replicates what the client currently has available to them and it was an easy way to ensure roles and security. It also instantly gave us versioning.

17.2 SharePoint Settings

The SharePoint was hosted on a private virtual server and was build with the default settings. The site chosen to host the web part was a document center, which is what gave us the most functionality that we needed (versioning and the like).

17.3 Web Part

The SharePoint web part built in C# was created in visual studio 2012 with default settings. Web parts were chosen over a SharePoint web app because it could be integrated easier into the virtual SharePoint and it was suggested by the clients as a solution to the web side of the requirements. Looking back, this was not the best choice because a web part does not have the flexibility of a web app, but it has the easiest setup.

Part VI
Results

We implemented a UI and a SharePoint web part.

18 UI

The client UI can be deployed with Newmont's standard software package. The client UI is populated from an xml document stored on the SharePoint, pulled down at startup. The client UI allows users to add metadata to files, this data is stored in an xml document that is stored on the SharePoint when the user uploads the file. When the file is uploaded the client program produces a zip folder containing all the selected files broken down by category. This zip is then uploaded to the SharePoint in the form of a stream due to size restrictions. This meets all of the client's requirements for the 'client UI' which included being able to dynamically generate the UI from xml, being able to edit metadata (comments and presets) about each file uploaded, being able to pull from mounted network drives and using their UNC path to reference them, and finally, being able to upload all of the files into a zip and

then storing them on the SharePoint. Due to time and security restrictions we were not able to implement some features including being able to pull from an already zipped master model file.

19 SharePoint Web Part

The SharePoint web part is a simple interface to see the data that had already been zipped together. It has a tree, metadata, and a button to download the zip file.

19.1 Deployment

Using the Visual Studio 2012 interface for SharePoint, we were able to deploy directly to the site without back-end configuration. The installation of visual studio had to be installed on the same machine as the SharePoint and then have an administrator for the site that was being deployed to sign in to that site using the Visual Studio application. Then it was just a matter of pressing Build > Deploy.