Project #4 – Remote Document Vault

due Monday, Nov 18th

Version 2.6

Purpose:

This project develops a Document Vault client and server. The purpose of the Vault is to enable insertion and extraction of text files to a remote location and to display information about their properties and relationships. One might expect that authenticated check-in and versioning would be required, but are not for this project due to the time available for its implementation.

The Vault client displays information about its file categories and files in a Graphical User Interface (GUI) with several panes used to display some metadata properties and text of the current categories and file. Clicking on a category or file makes that the current item.

For some applications document parent-child relationships are important, e.g., an Operational Concept Document has one or more Software Specification Documents (SSD)s as children. An SSD has a Software Design Document (SDD) as a child. Each SSD also has a Software Acceptance Test Document (SATD) child[1].

Child dependency relationships and file properties, including selected categories, are entered by the user when a metadata file is created. As part of this project you will also build a tool used to find the parent relationships for each file in the repository. The tool sequences through each metadata file and for every child records the current file as a parent in a map[2] used for navigation.

Requirements:

The Document Vault:

1. **Shall** be implemented in C# using the facilities of the .Net Framework Class Library (FCL), version 4.5, as provided in the EECS clusters. Use of a Graphical User Interface is required, and **shall** be implemented using Windows Presentation Foundation (WPF). Communication between Vault Client and the Document Vault **shall** be implemented using Windows Communication Foundation (WCF) and **shall** use message passing for all transactions between Client and Vault.

2. The Vault Client **Shall** support insertion and extraction of files to and from a, possibly remote, Vault Server using Windows Communication Foundation (WCF). Metadata for each stored file **shall** identify one or more user-defined categories. Categories are identified by metadata and **shall** be structured to support efficient queries[3].

3. **Shall** provide a separate tool that searches metadata files to build a map of parents for each file in the Vault. This tool is run each time the Repository starts. When a file is inserted into the Repository its parent relationship with each of its children is recorded in the map.

4. The Vault Client **shall** be a WPF application that provides an information view with two information panes, one for metadata and one for file text of the currently selected file. The client **shall** also provide two navigation panes, one showing the current file's children and one showing its parents. Clicking on either a child or parent changes the current file to the file represented by the clicked link. Note that these views are views of contents in the Document Vault[4].

---

[1] The intent of this project is to design so that users can decide how to structure the child relationships and what the relationships mean. The job of the Document Vault is to support those user needs.

[2] Here map simply means some kind of suitable container.

[3] An efficient query opens only those files that are in the categories specified in the query.

[4] You may wish to design the client to cache files and metadata locally to minimize network copies. If you do this your design should ensure cache coherency.

5. The Vault Client **shall** also supply a category view that displays all the categories in the Vault. Clicking on a category displays all the files in that category. Clicking on one of the category files makes that file the current file, subject to the requirements stated in the previous item. The Vault client **shall** provide a means to return to the category view at any time. The metadata for each file that has no file parents **shall** identify its categories as parents[5].

6. The Vault Client **shall** provide a second file management view that supports local browsing for files to insert into the remote Repository and means for the user to provide description text and lists of child dependencies and keywords and categories selected from those provided by the Document Vault. The metadata tool **shall** be used to automatically build a metadata file from the information supplied here. This view **shall** also support editing metadata information for the current file.

7. **Shall** require metadata construction for each document submitted for inclusion in the vault[6]. Thus every document contained in the vault will have a corresponding metadata file.

8. The Vault **shall** accept text query messages and return the fully qualified name[7] of every file containing that text in a clickable list. Text query searches **shall** be executed concurrently across files and accept one or more categories, one or more text strings to seek and a switch setting indicating whether matching any query string or all query strings is required.

9. **Shall** accept metadata query messages specifying one or more categories and metadata tags and return the contents of each element of the metadata for every file in the file set. A metadata query will always return its file reference and categories.

10. Clicking on any file in the text query results will make that file the current file and update views appropriately. Clicking on a file reference in metadata query results will make that file the current file and update client views appropriately.

11. **Shall** support editing metadata files[8] by supplying a set of tag names and values. If the tag exists in the metadata file being edited the value replaces the old value. If the tag does not exist in the edited file a new element is created with that tag name and value.

12. Both Document Vault and Vault Client **shall** implement as much of their functionality as practical in separately compiled components, e.g., a dynamic link library with interface and object factory.

13. Your submission **shall** supply a zip file containing the solution for your project and all its files. It **shall** also include compile.bat and run.bat files that compile and run your project without the user needing to specify any information. Please make sure these work as expected when you extract the zip in an empty directory at any level in your computer's file system. In order for that to work you must use relative directory paths in your Repository and Client code.

---

[5] Note that parent relationships are only held in the parent map, not in individual metadata files.
[6] Note that this requirement effects the design of both the Document Vault and its Clients.
[7] By "fully qualified" we mean file name and category, not necessarily a fully qualified path.
[8] There is no requirement to edit text documents.