**Project #4 – Remote Software Repository** due Tuesday, April 30 Version #1.2

Purpose:

This project requires you to build a Software Repository, a place to store and retrieve packages. The storage mechanism is based on the use of XML metadata to hold version and developer information as well as dependencies.   
  
Your repository will provide package check-in, display of Repository contents, and automatic versioning, according to rules specified below. You will also build a client that accesses the Repository functionality even if the Repository resides on another machine in the same network as the client.

Requirements:

Your Repository client and server programs:

1. **shall** use standard C++, the standard library, C++\CLI and Winforms or WPF, compile and link from the command line, using Visual Studio 2012, as provided in the ECS clusters and operate in the environment provided there[[1]](#footnote-1).
2. **shall** use services of the C++ std::iostream library and Win32 sockets for all input and output to the Repository server and .Net Winforms or WPF and sockets for all input and output to and from the Repository client. Both Repository client and server **shall** use the C++ operator new and delete for all dynamic memory management.
3. **shall** support user log-ins, accepting credentials and storing those in an XML file. You are not required to support encryption, as would be needed for a commercial implementation.
4. **shall** provide a check-in facility, using Win32 sockets for all client-server communication. This communication consists of passing messages from the client to server to initiate actions in the server, passing files between the client and server[[2]](#footnote-2) and sending notification messages from server to client indicating success or failure of the specified operation.
5. **shall**, for newly stored packages, create the package XML metadata and record the logged-in user name as the owner of the package and its version number as 1. For modified packages, e.g., a package exists in the repository with the same name, check-in shall fail if the logged-in user is not the package owner. If the logged-in user is the package owner, the package version number is incremented by one and stored in the repository, e.g., a metadata package file is created for that version, and the header and implementation files are stored in the repository.
6. Each version of each package **shall** have its own folder in the repository. For packages that have associated files, e.g., resources, stdafx.h, stdafx.cpp, etc. the Repository may store those files in the package folder[[3]](#footnote-3).
7. Check-in **shall** not be complete until all the dependencies for the package are defined by referring to packages already in the Repository and its file(s) are stored in the Repository. In this case we say the check-in is open. Another package in the Repository may refer to a package with open check-in. However the depending package check-in **shall** not be closed until its dependencies’ check-ins are all closed.
8. **Shall** support extraction by package name. An extraction specifies whether only the cited package is delivered to the client or the package and all its dependencies are delivered.
9. Because of the ownership policy, specified in 5, above, no check-out functionality **shall** be implemented. All retrieval is based on extraction, and any logged-in user may extract a package and use it in any way, including modification. But only the owner can check-in a modified package. Note that this does not preclude a user from extracting a package owned by another developer, modifying it, and checking it in as the owner under a new name.
10. **shall** provide a client display of the Repository contents. The display **shall** provide means to navigate through the Repository’s dependency structure[[4]](#footnote-4).
11. **shall** be submitted containing all packages from your Repository client and server, along with corresponding metadata.
12. Your project submission **shall** be uploaded in a zip file archive, including batch files named compile.bat and run.bat that compile and run your project using appropriate command line arguments. Please also include a Visual Studio solution that demonstrates you meet these requirements.

1. VC++ version 8.0 is provided by Visual Studio .Net 2005, and is available in all the ECS clusters. [↑](#footnote-ref-1)
2. You are required to support file transfer in either direction. [↑](#footnote-ref-2)
3. This will be particularly useful for GUI interface packages that have a lot of associated files. [↑](#footnote-ref-3)
4. We will discuss ways to do this in class. [↑](#footnote-ref-4)