**Project #4 - Dependency-Based Remote Code Repository**

Version 1.3, Revised: 04/12/2016 05:13:05  
Due Date: Tuesday May 3rd

**Purpose:**

A Code Repository is a Program responsible for managing source code resources, e.g., files and documents. A fully developed Repository will support file persistance, managment of versions, and the acquisition and publication of source and document files. A Remote Repository adds the capability to access the Repository's functionality over a communication channel, e.g., interprocess communication, inter-network communication, and communication across the internet.

In this project we will develop a dependency-based code repository and communication channel that supports client access to the repository from any internet enabled processor.

The communication channel will use sockets and support an HTTP like message structure. The channel will support:

* HTTP style request/response transactions
* One-way communication, allowing asynchronous messaging between any two endpoints that are capable of listening for connection requests and connecting to a remote listener.
* Transmission of byte streams that are set up with one or more negotiation messages followed by transmission of a stream of bytes of specified stream size1.

The Code Repository will:

* Support either module or package checkin, where a module is a named small set of closely related packages, usually from the same namespace. You only have to support one of these.
* Store files in folders based on module namespace or package name and checkin date2.
* Track dependency relationships between modules or packages.
* Support extraction of all the files in some dependency sub-graph by naming only the root module or package of the sub-graph.
* You may, but are not required to evaluate dependency relationships during check-in, using analysis facilities you developed in previous projects this semester.

You will track dependencies and versions using metadata files with XML contents. Alternately you could use a NoSql Database like that developed in CSE681-Software Modeling & Analysis, Project #2, Fall 20153. However, that would require porting to C++ and would contribute a lot of additional work needed for this project.

**Requirements:**

Your Remote Repository4:

1. **Shall** use Visual Studio 2015 and its C++ Windows console projects, as provided in the ECS computer labs. You must also use Windows Presentation Foundation (WPF) to provide a required client Graphical User Interface (GUI).
2. **Shall** use the C++ standard library's streams for all console I/O and new and delete for all heap-based memory management5.
3. (**2-4**) **Shall** provide a Repository program that provides a module and/or package checkin process. Each checkin will support copying and providing metadata for all modules or packages in the Repository. Metadata contains a description of the module/package and a list of other modules/packages on which it depends.
4. (**4**) **Optionally** automate checkin dependency analysis using the facilities you developed in Project #3.
5. (**3-6**) Each module or package checkin **Shall** create a new directory with a unique name based on the module or package name and checkin-date. Each module directory contains module metadata containing a description of all packages and package metadata that contains a list of all the files in the package.
6. (**2**) Each checkin directory **Shall** have a boolean closed property. If the directory is not closed files may be added or changed at any time without changing the directory's date. Once closed the directory contents become immutable.
7. (**4-6**) **Shall** provide an extraction process for modules or packages with or without dependencies.
8. (**3**) **Shall** provide a message-passing communication system used to access the Repository's functionality from another process or machine.
9. (**3**) The communication system **shall** provide support for passing HTTP style messages using either synchronous request/response or asynchronous one-way messaging.
10. (**3**) The communication system **shall** also support sending and receiving streams of bytes6. Streams will be established with an initial exchange of messages.
11. (**5**) **Shall** include an automated unit test suite that demonstrates you meet all the requirements of this project7 including the transmission of files.
12. The stream capablity is intended to send files, which could be either text or binary format. Stream size will be the file size.
13. This Repository uses module or package names and dates to define unique versions.
14. You could do this either by porting the NoSql database to C++ or by building a managed C++ wrapper around the C# NoSql database.
15. Note that the requirements score can be more than the usual 25 points. The intent is that you have some flexibility in what you implement and, for those of you with enough time, the possibility of scoring more than 100 points on this Project.
16. That means that you are not allowed to use any of the C language I/0, e.g., printf, scanf, etc, nor the C memory management, e.g., calloc, malloc, or free.
17. Transmitting and receiving byte streams will be used to send and receive files in either text or binary format.
18. This is in addition to the construction tests you include as part of every package you submit.