Bonus Prototypes

1. Repository manifest builder and scanner with tests showing how it works.
2. Wizard to quickly build source code for test packages – inputs are name, name of production code package(s). Uses ITest and AbstractTest (see midterm solutions) and default TVG and Logger.
3. Test Suite builder that uses csc to compile and build test libraries. Required to include timing, using high resolution timer, and timed results in a graph that shows time to compile versus lines of code.
4. WPF Client GUI that mocks up client operation and illustrates how notification will work.
5. Communication system with a service contract that has asynchronous PostMessage and streaming file download operation contracts, e.g., pushes messages and pulls files.
6. Package authenticator that hashes a source file's checksum, embeds it as a comment in the file, and authenticates by checking the hashed checksum with a computed checksum (after removing the comment). It is acceptable to substitute some other identifier for the checksum if you can think of a good one.
7. Write code for a version manager that accepts an unversioned file and converts it to a versioned file. The manager should also accept a versioned file and increment its version number. You may version by name or by directory. However, the version manager should also insert a version number comment at the top of each accepted source file. Examine every file generated for a a) C# console application, and b) a C# WPF application. How might you manage versioning of the additional files you find? Do you need to do that?

Required Prototypes are listed toward the end of the Project #5 Statement.