**Project #1 – Test Generator** due Tuesday, Feb 07 version 2.1

Purpose:

When projects get large, with perhaps hundreds of packages, code changes frequently because of changing requirements, latent errors, performance problems, or because child packages[[1]](#footnote-1) change. For each of these the changed package must be retested, and it is advisable to retest all of the packages that directly depend on a changed package. So, when a segment of the code baseline is under active development it is likely that most of the segment’s packages will be tested many times. For this reason we need to run automated tests in a test harness.

In this project we will begin the development of a test harness facility which will be completed over all of the projects we will work on this semester. This project focuses on building test libraries and providing support for supplying test inputs and logging test results. Each library is composed of the package being tested, any packages on which it depends[[2]](#footnote-2), and a test driver that implements an ITest interface consisting of the methods static ITest\* create() and bool test()[[3]](#footnote-3). It also contains a supplier of test inputs, TestDataGenerator, that implements an ITestGenerator interface, and a logging facility that implements an ITestLogger interface.

This OOD project requires you to build a test wizard that accepts an XML file containing a test Name and a list of the files to test. It then generates the files: NameTestDriver, NameTDG, NameTL, and a Visual Studio console project that includes the three files you generated and all the tested files, Name.vcxproj.

Requirements:

Your TestGenerator Wizard:

1. **shall** use standard C++[[4]](#footnote-4) and the standard library, compile and link from the command line, using Visual Studio 2010’s C++ compiler, as provided in the ECS clusters and operate in the environment provided there[[5]](#footnote-5).
2. **shall** use services of the C++ std::iostream library for all input and output to and from the user’s console and C++ operator new and delete for all dynamic memory management.
3. **shall** accept from the command line a specification of the path to a directory containing the XML test configuration file, interfaces, and tested code[[6]](#footnote-6).
4. The XML file **shall** provide a test name and a list of files to test. Optionally, it may include a path to find the required interfaces instead of placing them in the same directory as the tested files.
5. **shall**, if no XML file is found, create a test name and assume that all the source code files in the specified directory are to be tested and that the interfaces are found in the specified directory[[7]](#footnote-7).
6. **Shall** generate three packages, a test driver and test data generator and logger, both used by the test driver. These files shall implement the ITest, ITestGenerator, and ITestLogger interfaces, respectively. These are not expected to contain test code, but rather provide functions that are empty or nearly empty for the user to fill in, just like the other Visual Studio wizards.
7. **Shall** generate a Visual Studio project[[8]](#footnote-8) that will compile the generated and tested files into a library file.
8. **Shall** provide a console application that loads the test library, creates its instances, and invokes its test.
9. **Shall** provide a display, using the facilities of the logger, to provide the test results to the user in a console window.
10. **Shall** use these facilities to test the Tokenizer package provided here:
<http://www.lcs.syr.edu/faculty/fawcett/handouts/CSE687/code/Parser/>

Note that there is no requirement to provide a graphical user interface. If you do so, you should also provide a command line interface, as required in #3. In order to successfully complete this project you will need to write code that:

* Reads and parses XML: <http://www.lcs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/Cpp/Code/XmlReader/>
* Loads and Uses DLLs: <http://www.lcs.syr.edu/faculty/fawcett/handouts/CSE687/code/DLL_ProtocolDemo/>
* Read and write files using the facilities of std::fstream
* Manipulate strings using std::string

You will also need to understand a little bit of what the Tokenizer package does. We will discuss that in class.

1. Child packages are children of a specified node in the project’s dependency graph. [↑](#footnote-ref-1)
2. Dependent code may be included statically or as a dynamic link library. [↑](#footnote-ref-2)
3. The create function is a factory method that creates instances of whatever classes are needed to execute the test method. [↑](#footnote-ref-3)
4. This means, for example that you may not use the .Net managed extensions to C++. [↑](#footnote-ref-4)
5. Visual Studio 2010 is available in all the ECS clusters, and can be downloaded at no cost from the MSDN Academic Alliance. [↑](#footnote-ref-5)
6. You will place the interfaces and tested code in that directory and write the wizard output to that same directory. [↑](#footnote-ref-6)
7. It is acceptable to specify a path to the interfaces as a defined constant string in the code for the wizard, to be used as a default location that can be overridden by the XML configuration file. [↑](#footnote-ref-7)
8. You may wish to generate a sample project using Visual Studio and then edit it to become a template for your wizard. [↑](#footnote-ref-8)