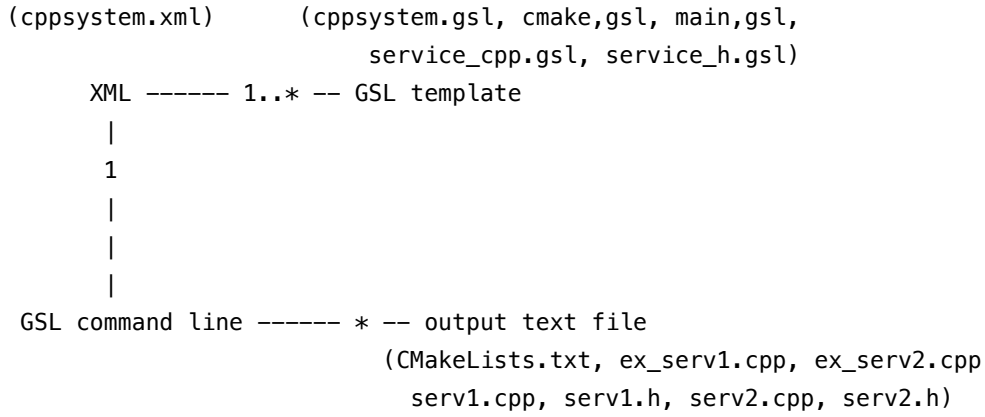
gsl_example

If you are looking for usage examples of the FOR&IF control flow statement of iMatix GSL code generator, I created a simplified C program functions model in XML and use GSL to generate C header and source files.

GSL can be found here https://github.com/imatix/gsl (1)
GSL specification can be found here https://imatix-legacy.github.io/gslgen/gsldoc4.htm (2)

Demo

The GSL application concept for the demo



GSL command line program

You will have to install the GSL command line program. GSL installation guide can be found here https://github.com/imatix/gsl

What I have on my computer

```
$ gsl
GSL/4.1.4 Copyright (c) 1996-2016 iMatix Corporation
syntax: gsl -<option> ... -<attr>[:<value>] ... <filename> ...
.. more detail about the command-line options omitted
```

CMake (https://cmake.org/) is required if you want to compile the example C code files demonstrated by this project

Generate code

```
$ git clone https://github.com/vorachet/gsl_example.git
$ cd gsl_example
$ mkdir generated
$ gsl cppsystem
```

Once the command `gsl cppsystem` can be executed successfully, generated code files will be saved in the ```generated`` folder

```
gsl_example/generated
        ├── CMakeLists.txt
        ├── ex_serv1.cpp
        ├── ex_serv2.cpp
        ├── serv1.cpp
        ├── serv1.h
        ├── serv2.cpp
        └── serv2.h
```

Compile code

This step is optional. It deonstrates the creation of `CMakeLists.txt` and how to create executable C programs (`ex_serv1` , `ex_serv2`) from the example project.

```
$ cd generated
$ mkdir build
$ cd build
$ cmake ..
$ make
```

```
gsl_example/generated/build
        ├── CMakeCache.txt
        ├── CMakeFiles/ many files here
        ├── Makefile
        ├── cmake_install.cmake
        ├── ex_serv1
        └── ex_serv2
```

What I've learned

Advantages of GSL

- Super Fast
- Tiny program (<1Mb)
- Good Documentation
- No need to develop metamodels
- Built-in XML-based metamodel parser
- No IDE/Workbech/GUI required

GSL can be one of your productive software engineering tools. GSL has developed by Pieter. (R.I.P Pieter Hintjens - the creator of ZMQ and GSL). Pieter contributed many productive works to the software industry.

I am still looking for a solution for the following GSL application issues

- Issue-1 Merge task - How to use GSL to preserve manual code modifications across re-generations ?
- Issue-2 To parse the `attr2` tag (XML element rather than XML attribute) with the following example XML model seems not possible with GSL?

```
<concept attr1="value1">
   <attr2>value2</attr2>
</concept>
```

As an MDE learner who uses more than one code generation tool, the following notes may be useful if you are going to fix the issues and don't mind using alternative tools differnt approach. Eclipse Epsilon (https://www.eclipse.org/epsilon) The Epsilon has features to satsify the issues. Issue-1 can be satisfied by https://www.eclipse.org/epsilon/doc/egl/#merge-engine. Issue-2 can be satisfied by the Eclipse Flexmi https://www.eclipse.org/epsilon/doc/flexmi/

I very much like a shell based environment and often perform MDE tasks with shell scripts. I developed `mdeshell` , a bash shell script based modeling environment that facilitates the definition of XML-based EMF metamodels/models and the execution of Model-2-Text transformation based on Model-Driven Engineering system implemented by the Eclipse Emfatic and the Epsilon Flexmi/EGL/EGX. Find more detail here https://github.com/vorachet/mdeshell