SIDEREUS - FRANCESCO SERAFIN

# GROW WORKING HARD

GROWWORKINGHARD.ALTERVISTA.ORG

THE PUBLIC IS MORE FAMILIAR WITH BAD DESIGN THAN GOOD DESIGN. IT IS, IN EFFECT, CONDITIONED TO PREFER BAD DESIGN, BECAUSE THAT IS WHAT IT LIVES WITH. THE NEW BECOMES THREATENING, THE OLD REASSURING.

PAUL RAND

A DESIGNER KNOWS THAT HE HAS ACHIEVED PERFECTION NOT WHEN THERE IS NOTHING LEFT TO ADD, BUT WHEN THERE IS NOTHING LEFT TO TAKE AWAY.

ANTOINE DE SAINT-EXUPÉRY

. . . THE DESIGNER OF A NEW SYSTEM MUST NOT ONLY BE THE IMPLEMENTOR AND THE FIRST LARGE-SCALE USER; THE DESIGNER SHOULD ALSO WRITE THE FIRST USER MANUAL. . . IF I HAD NOT PARTICIPATED FULLY IN ALL THESE ACTIVITIES, LITERALLY HUNDREDS OF IMPROVEMENTS WOULD NEVER HAVE BEEN MADE, BECAUSE I WOULD NEVER HAVE THOUGHT OF THEM OR PERCEIVED WHY THEY WERE IMPORTANT.

DONALD E. KNUTH

# *Grow Working Hard*[1]

*sidereus - Francesco Serafin*

*15 May 2015*

[1] Inspired by Jonathan R. Shewchuk

> Everyday life is like programming, I guess. If you love something you can put beauty into it.
>
> - Donald Knuth -

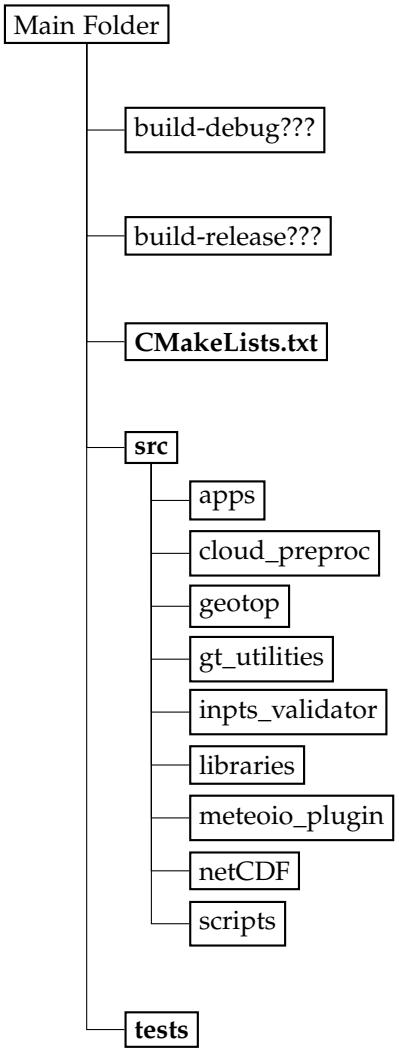
This document describes my daily experiences on Hydroinformatics and IT. It provides recipes followed to reach each and every small goal. The main purpose pursuaded is the redesign of the famous hydrological model GEOtop.

GEOtop is a distributed model of the mass and energy balance of the hydrological cycle, which is applicable to simulations in continuum in small catchments. GEOtop deals with the effects of topography on the interaction between energy balance and hydrological cycle with peculiar solutions.

## *Jun 24th 2015 - GEOtop package: the initial idea*

### *Changing the structure of the folders*

The main folder will be structured as follow:




I think this work is necessary. Currently, the structure is really a mess: the folders of the source code are mixed with the tests folder and many different files are located in the main working directory, e.g. COPYING, INSTALL, NEWS which might be gathered in a **doc** folder.

Cleaning up and tyding up are necessary.

## *Jul 7th 2015 - Research proposal*

Before starting to write the complete research proposal, I need a cheat sheet.

### *Positive aspects in GEOtop nowadays*

1. Implicit numerical method;
2. Resolution of 1D and 3D problems;
3. Complex system to compute the solar height;
4. Complex system to compute the snow fallen and its compacting;
5. Resolution

### *Negative aspects in GEOtop nowadays*

1. Structured mesh;
2. Monolithic and messy code;
3. Quite impossible adding new equation to solve the same problem in a different way;
4. Poor memory management (fluidturtle starting from 1 instead of 0);
5. So that MeteoIO could "talk" with GEOtop, it's necessary copying each matrix in input value by value;

## *Jul 9th 2015 - GEOtop CMake - Redesign*

Today I started working on the CMake structure of the GEOtop source code. A fork of the official GitHub repository has been done, in order to feel free to try to develop the new building-system. The forked repo is `https://github.com/francescoS/geotop`.

A new branch in my own forked repo has been created, called **cmake_redesign**.

First of all, the new folder structure has been applied. Then, the main CMakeLists.txt has been splitted creating a new one inside the src folder[2].

In the main CMakeLists has been added the target install, in order to install the header files in the include directory. Problem: some folders (app, cloud_preproc) don't contain header files. Is it necessary to write the corresponding header file?

Following some notes about the commands of CMake scripting language. These notes are taken from CMake Tutorial (web-site version).

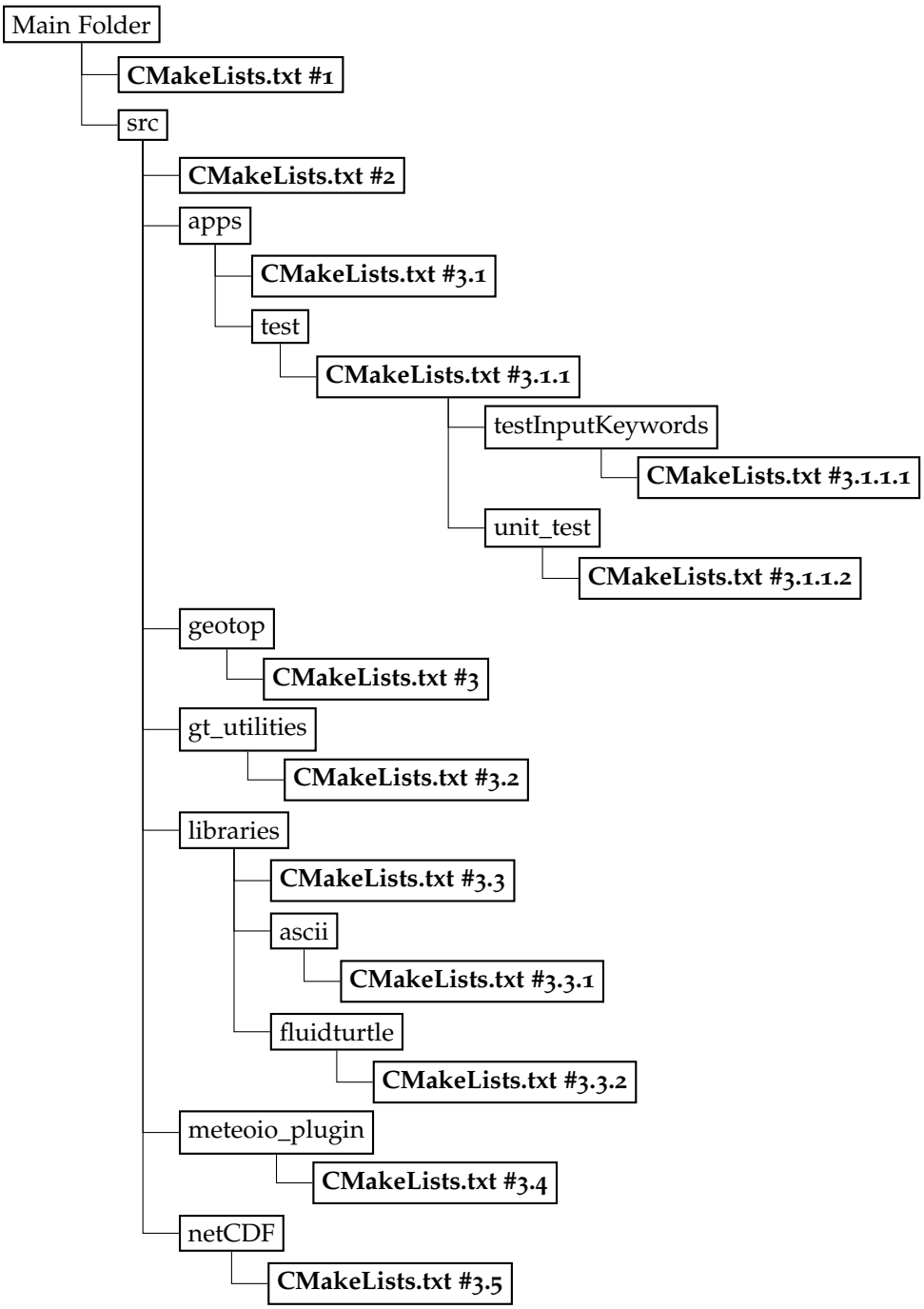
[2] Add description about the two main CMakeLists.

```
1 ADD_EXECUTABLE(PRJ_NAME executable.cxx) # I dont'know if I'm obliged to use the PRJ_NAME to add the executable. More
      research are necessary.
2 INCLUDE_DIRECTORIES(''DIR_PATH/DIR_NAME'') # so that the header file can be found for function prototype
3 ADD_SUBDIRECTORY(DIR_NAME) # to make use of the new library, so that the library will get built
```

*Jul 14th 2015 - GEOtop: the new building-system*

On Saturday 11$^{th}$ of July, 2015 the new **GEOtop Building-System** had been completed. The code is on the branch **cmake_redesign** of the forked repository `https://github.com/francescoS/geotop.git`.

The new CMakeLists structure is the following:



*CMakeLists.txt #1*

This is the redesign of the original main CMakeLists in GEOtop. The previous one was very long and complex because of the necessity to setting up the different variables (TARGET, LIBRARY_PATH, INCLUDE_PATH) for every library to link to the executable. This was due to the fact that in the main folder there were library folders, test folder, source folder, etc indistinctly.

The new structure was necessary mainly for two reasons:

- ORDERLINESS, in order to have the source code in a directory, while documentation and tests are in their own directories;
- BUILDING-SYSTEM SCRIPT READABILITY, because of the add of
  - *install target* to install the header files, executable and linked libraries;
  - *custom target* to get a working make clean;
  - *uninstall target* to uninstall the header files, the executable and the linked libraries.

  the main CMakeLists would have become too long, difficult to read, maintain and develop.

Now the main CMakeLists contains only generic build-system settings, divided in 8 sections (following listed and explained):

1. Intro section with
   - default compiler flags;
   - project name;
   - source code version and setting of the config.h file
2. User options
3. Find dependencies
4. Libraries linking type and extensions
5. Extra compiler flags
6. Install target
7. Clean target
8. Uninstall target

TODO => Implementation of the MSVC variable, in order to allow the compilation through Windows Visual Studio.

## *Jul 15th 2015 - GEOtop: the new building-system*

### *CMakeLists.txt #2*

This cmake file is located inside the src directory and it is simply a part of the previous main CMakeLists. It contains the variables set up to link the compiled libraries to the executable.

### *CMakeLists.txt #3*

In this file, every library is simply linked to the executable. The target install part has been added immediatly after the linking target of every library, in order to install it.

### *CMakeLists.txt #3.1 - #3.5*

In all the other CMakeLists, the target install has been added.

### *Jul 16th 2015 - GEOtop building-system summarize*

The main development points are:

1. Realize the new CMakeLists structure;
2. Write the code to compile GEOtop with
   - OSX;
   - Windows.
3. Create Virtual Machines for several GNU/Linux OS to compile and create .deb packages every time the GEOtop source code is modified. Eventually, create a bash script in order to automate this process
4. Packages testing
5. Create packages with automatic dependencies resolution

Trying to create a .deb package to solve automatically all the dependencies, I found out this usefull link:
`https://wiki.debian.org/IntroDebianPackaging`
Dependencies to solve (Debian OS) to compile MeteoIO 2.4.2:

- libproj-dev

ATTENTION!!! Compiling MeteoIO 2.4.2 on Debian stable 8.1 there is a compatibility dependency problem with Ubuntu 14.04: the last one has gcc 4.8.4 in the default repositories, while debian is using gcc 4.9. The only solution is compile the code, also GEOtop source, in ubuntu and testing it on Debian.

Problem solved adding a new repo in ubuntu:

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt-get install gcc-4.9
```

After installing the right version of gcc, the installation of the precompiled package meteoio_2.4.2-1_i386.deb has been done through GUI solving the **libproj** dependecies automatically, just adding the root's password in order to install meteoio.

Anyway, this solution could be a problem for the normal user who should struggle with command-line and terminal.

### *Aug 4th 2015 - GEOtop building on Windows 7 with Cygwin*

Since yesterday, the main goal in the following few days is building GEOtop under Windows 7.
Everything is totally new. The source codes are going to be built as static libraries in order to get a complete executable, maybe exploitable in every machine with a Win7 OS.

The following tools have been installed:

- Cygwin (`https://cygwin.com/install.html`). During the installation process select:
  - *Development* option;
  - at least one terminal editor;
- Proj4 (`https://github.com/OSGeo/proj.4.git`
  git clone https://github.com/OSGeo/proj.4.git):
  - First of all, there is no direct support for Cygwin. So switch branch to git checkout tags/4.7.0 and apply the patch (`https://github.com/minitage-dependencies/proj-4/blob/master/cygwin.patch`). Actually, the building of Proj4 on Cygwin ends without errors or problems, however it doesn't properly work also with MeteoIO examples, due to the lack of correct flags to build the system under Cygwin;
  - the version 4.7.0 of the code has no support for CMake, so it has to be build with configure

```
# building static libraries
[fra@sidereus] ~ $ ./configure --enable-shared=off
[fra@sidereus] ~ $ make
[fra@sidereus] ~ $ make install
```

- MeteoIO 2.4.2 (`http://models.slf.ch/p/meteoio/downloads/220/`);
  - build static ON;
  - build shared OFF;
  - Proj support ON;

  with the last option the find_package() of CMake should recognize the libproj.a.
  Building the MeteoIO examples with static libraries I get the following error:

```
/usr/local/lib/libmeteoio.a(Coords.cc.o): In function `ZNK3mio6Coords14WGS84_to_PROJ4EddRdS1_':
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1480: undefined reference to `init_plus'
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1484: undefined reference to `pj_init_plus
    '
```

```
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1490: undefined reference to ‘pj_transform’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1498: undefined reference to ‘pj_free’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1499: undefined reference to ‘pj_free’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1485: undefined reference to ‘pj_free’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1486: undefined reference to ‘pj_free’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1481: undefined reference to ‘pj_free’
/usr/local/lib/libmeteoio.a(Coords.cc.o):/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1492: more undefined references to ‘pj_free’ follow
/usr/local/lib/libmeteoio.a(Coords.cc.o): In function ‘ZNK3mio6Coords14PROJ4_to_WGS84EddRdS1_’:
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1524: undefined reference to ‘pj_init_plus’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1528: undefined reference to ‘pj_init_plus’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1534: undefined reference to ‘pj_transform’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1542: undefined reference to ‘pj_free’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1543: undefined reference to ‘pj_free’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1525: undefined reference to ‘pj_free’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1536: undefined reference to ‘pj_free’
/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1537: undefined reference to ‘pj_free’
/usr/local/lib/libmeteoio.a(Coords.cc.o):/cygdrive/c/Users/francesco/source_code/MeteoIO-2.4.2/meteoio/Coords.cc:1529: more undefined references to ‘pj_free’ follow
collect2: error: ld returned 1 exit status
CMakeFiles/2D_interpolations.dir/build.make:86: recipe for target ’2D_interpolations.exe’ failed
make[2]: *** [2D_interpolations.exe] Error 1
CMakeFiles/Makefile2:60: recipe for target ’CMakeFiles/2D_interpolations.dir/all’ failed
make[1]: *** [CMakeFiles/2D_interpolations.dir/all] Error 2
Makefile:76: recipe for target ’all’ failed
make: *** [all] Error 2
```

Why? I have got no answer. Anyway, just linking the Proj4 library again in the CMakeList.txt of the MeteoIO examples solves the problem.

- Boost (`http://sourceforge.net/projects/boost/files/boost/1.58.0/`);

```
[fra@sidereus] ~ $ ./bootstrap.sh --with-libraries=filesystem,system,iostreams,regex,program_options,test
[fra@sidereus] ~ $ ./bjam link=static install
```

### *Aug 17th 2015 - GEOtop exe on Windows 7 (with Cygwin)*

Today I'm writing the recipe to use geotop-2.0.1.exe.

1. Cygwin (`https://cygwin.com/install.html`). During the installation process select the following item in the **Development** option:

   - **autoconf2.5**
   - **cmake**, **cmake-gui**
   - **gcc-core**
   - **gcc-g++**
   - **git**
   - **make**
   - **patch**

### *Aug 18th 2015 - Install Docker on ArchLinux*

To install Docker on Arch Linux, run the following commands:

```
$ yaourt -Syu
$ yaourt -Ss docker
$ yaourt -S docker
$ sudo ln -sf /usr/bin/docker /usr/local/bin/docker
```

and then, if you are a bash user, the docker completaion has been already set up.

*Aug 24th 2015 - Building GEOtop on OS X*

https://growworkinghard.wordpress.com/2015/08/26/building-geotop-on-os-x/

*Aug 25th 2015 - Building GEOtop on Windows 7*

https://growworkinghard.wordpress.com/2015/12/17/building-geotop-on-windows-7/

### *Sep 26th 2015 - Setting up a Java environment*

Today I just simply set up a Java IDE for vim. Why? I tried using both Eclipse and Netbeans, but these are not editor for me.

It's quite impossible to set up a working Java IDE on Emacs. So I moved to vim because it works very well if coupled with **eclim** (the **eclimd** daemon has to be manually run every time you need it in a vim session by `$ECLIPSE_HOME/eclim`. You can the stop it inside vim with `:ShutdownEclim` or with `$ECLIPSE_HOME/eclim -command shutdown`).

The study of the Strategy Pattern has finished.

*Sep 28th 2015 - Design Patterns*

Going on studying Design Patterns, yesterday and today I finally got the following patterns:

- **Strategy Pattern**: it allows the developer to separate and encapsulate what varies from what remains the same;
- **Observer Pattern**: it allows to create a relationship between an Observable which send data (push or pull methodology) to many Observer, knowing nothing about the Observers. This means that there is a loosely coupled design between objects that interact;
- **Decorator Pattern**: considering the chance to wrap and object with many different type of decorators, this pattern allows that opening the main class for extension, but closing it for modification;
- **Factory Method Pattern**: encapsulate the object creation in subclasses, delegating them to decide which class to instantiate;
- **Abstract Factory Pattern**: provides an interface for creating families of related or dependent objects without specifying thei concrete classes;
- **Singleton Pattern**: allows the developer having to deal with only one istance of that class, providing a global point of access to it.

I would like to apply the Factory Method Pattern on Marialaura's components. Specifically, I think this pattern might properly fit the issue of manage more than one numerical solver, choosing it at run time.

Discussed with Marialaura about managing her components in OMS (the code for computing the Residence Time is defintely too slow): unfortunately no solution has been found because the problem is due to the fact that OMS and consequently reader and writer work timestep per timestep.

### *Sep 29th 2015 - Design Patterns*

Implementation and studying of the **Command Pattern** and of

- null object;
- macro;
- lamba expression (function object);
- method reference.

Reading page 237 of Head First Design Patterns, why doesn't manage the single branch of the net (the single HRU) as a queue? You could send to OMS branch by branch (considering the Pfafstetter numbering) solving it in queue. In this way it's possible to implement the managing of the net through the **Command Pattern**.

I start reading **Algorithms**.

### *Sep 30th 2015 - Design Patterns*

Carrying on with Design Patterns today I learnt how to implement:

- **Adapter Pattern**: it converts the interface of a class into another interface the client expect. There are two Adapter Patterns:
  - **Object Adapter**: which uses object composition to wrap the adaptee with the altered interface. It works on Java;
  - **Class Adapter**: which uses inheritance to wrap the adaptee with the altered interface, because it needs multiple inheritance to be implemented. It doesn't work on Java.
- **Facade Pattern**: it defines a high-leve interface that makes the subsystem easier to use. In this way it decouples a client from a complex subsystem.
- **Template Method Pattern**: it allows to manage an algorithm (defining a skeleton of it in the Template Method) in an abstract class, delegating to subclasses the implementation of some steps of the algorithm. Subclasses can redefine certain steps of an algorithm without changing the algorithm's structure. In the algorithm there might be one or more hooks, methods that are declared in the abstract class, but only given an empty or default implementation. This gives subclasses the ability to "hook into" the algorithm at various point, if they wish; a subclass is also free to ignore the hook.
- **Iterator Pattern**: it provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation. You might have:
  - *External Iterator*: the client controls the iteration bu calling `next()` to get the next element;
  - *Internal Iterator*: it's controlled by the iterator itself which steps through the elements, so that you have to tell the iterator what to do with those elements as it goes through them.

*Oct 1st 2015 - Design Patterns*

- **Composite Pattern**: allows to composite objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
- **Composite Iterator**: allows to traverse the tree structure analizing leaf or composite and making sure all the child composites (and child child composites, and so on) are included.

## *Oct 5th 2015 - Algorithm*

Discussing with Marialaura about the schematization of the river Adige and after having read Wang et al. (2011) [3], we arrived at the following ideas:

- Focus on **OPN** (Optimal Processor Number): this is going to be the innovation of this work;
- Meteo stations will be treated as input data, at the moment I cannot see a way to include them in the tree;
- The Net (and the binary tree) will be composed by:
  - streams;
  - dams;
  - lakes;
  - hydrometers.

In order to minimize the users's efforts, just the shapefile of stream has to be required in order to process the net. The dbf is going to be read and probably only the following columns are necessary:

- Hack numbering to get the last stream (to the closure);
- Coordinates of the starting point of the single stream;
- Coordinates of the ending point of the single stream.

With these informations I should be able to build the tree just looking at the coordinates of the interconnection points. If a node has more than two children a "ghost" node is introduced, following Wang et al. (2011) [4]. The operation of building the tree, can be parallelized, because the numbering of each child depends only on the number of the parent node. Each branch of the tree is independent.

The idea is building the tree into an arraylist for example, managing a key and an object following the *Composite Pattern* which can be:

- a complete node;
- a ghost node;
- a dam;
- a leaf.

The structure of the Composite Pattern should help in realizing a flexible framework. To get the data about dams, another dbf could be read with coordinates of the dams.

At the moment, no idea about lakes.

[3] Wang, H., Fu, X., Wang, G., Li, T., and Gao, J. (2011). A common parallel computing framework for modeling hydrological processes of river basins. *Parallel Computing*, 37(6):302–315

[4] Wang, H., Fu, X., Wang, G., Li, T., and Gao, J. (2011). A common parallel computing framework for modeling hydrological processes of river basins. *Parallel Computing*, 37(6):302–315

## *References*

Wang, H., Fu, X., Wang, G., Li, T., and Gao, J. (2011). A common parallel computing framework for modeling hydrological processes of river basins. *Parallel Computing*, 37(6):302–315.

*Oct 9th 2015 - Multithreading*

> There's ways to amuse yourself while doing things and thats how I look at efficiency.
>
> — Donald Knuth

*Oct 14th 2015 - RiverNe3*

My general working style is to write everything first with pencil and paper, sitting beside a big wastebasket. The I use Emacs to enter the text into my machine.

— Donald Knuth

The most important thing in the kitchen is the waste paper basket and it needs to be centrally located

— Donald Knuth

Playfully doing something difficult, whether useful or not, that is hacking

— Richard M. Stallman

*Sep 20th 2016 - Back to life*

Back writing after some time.

Today's work: well, I started to take classes of Algorithms and Data Structures at DISI. The purpose is to have a background before taking classes at the CS Department at CSU.

In the meanwhile, my main goal is the implementation of a Tree data structure inside OMS core. Olaf allowed me to push directly on a forked version of the OMS core on `http://alm.engr.colostate.edu`. The starting point of the analisys is the `.sim` file to understand how to connect the OMS core to the user experience. A new thread is created at row 770 in `SimPanel.java` after clicking the `runButton`, thus the parsing of the `.sim` file should be done more or less in this part of the source code. A `ProcessExcecution` starts at line 783, calling the method `exec()` at row 205 in `ProcessExecution.java` class.

Furthermore, I completed the work on the building script of the `https://github.com/geoframecomponents/GEOframeUtils` repository. The building system is Gradle[5] and the dependencies management is automatically done through maven repositories. For this reason, a **maven repo should be organized** to put the `jars` of each component for who is going to use them as API.

5

**FIXED UGLY FONT RENDERING** in Netbeans on ArchLinux. The trick is running `Netbeans` with the following options

```
1  netbeans -J-Dswing.aatext=true -J-Dawt.useSystemAAFontSettings=on
```

### *Sep 27th 2016 - Tree ds inside OMS*

After few days vacation, I came back to work. This morning I firstly ended to set up the page style of the logbook. I upgrade the macOS Yosemite to Sierra, fixing a problem due to a too strict management of privileges.

On Tuesday the 20th of September I started taking class of Algorithm and Data Structures. It is really necessary to implement clean and fast algorithms. Today class 2pm - 4pm: exercises about **recurrence equations**.

I still have to implement the Tree data structure inside OMS. The main problem is to find the correct place to put the source code to read the links between nodes and populate the data structure. I'm working on the `SimPanel.java` class, reading, cleaning and commenting the code, hoping to find where the `.sim` file is parsed. I would like to add a section where reading the tree populating the data structure. I wrote a mail to Olaf asking for a help.