**Intro**

This is a project plan for migrating unstructured grid editing functionality from Fortran to C++ and adding the support of CGAL geometrical library for performing low-level geometrical operations (e.g. inquiring segment intersections, distances, and directions). Most of the algorithms are contained in the files net.f90 and unstruct.f90 of the DFlowFM source code. Examples of grid operations are mesh orthogonalization and curvilinear grid generation from splines.

Our approach to expose the algorithms to the front end (the Grid Editor) has been to wrap the Fortran code in separate API calls. Although this approach immediately delivers functionality, it has several drawbacks:

1. The code was not designed beforehand for interacting with the Grid Editor. Visualizing the mesh during the computations requires a major refactoring of the algorithms.
2. The Fortran code style is not self-explanatory (e.g. names of variables are often an abbreviation of Dutch names, e.g. KN, LN, LNE) and sometimes implementations are repeated (e.g. the algorithm to find the cells from the edge table is copied 4 times with different levels of nesting).
3. The variables used by the algorithms are contained in several independent data modules. When several modules are used in the same subroutines, hundreds of variables are imported in the local scope and it is not clear which variable is modified unless the function implementation is followed line by line. In subsequent functions the same modules can be imported again and the result of the computation is difficult to predict. This approach breaks the encapsulation principle (gluing together data and related methods in one object) and makes the use of the library outside the interactor environment brittle.
4. Using Fortran does not allow to leverage productivity tools such as ReSharper. Currently browsing and writing DFlowFM code in Visual Studio 2015 is very slow and poses a serious productivity issue.

This project aims to overcome the drawbacks listed above by rewriting the basic mesh administration routines and the orthogonalization function in C++ and by employing the CGAL library for low-level geometrical operations (see Figure 1).

Here how the points listed above will be addressed:

1. The mesh orthogonalization function will be separated into three functions (initialization, iteration, and finalization). The client will be able to control the outer iterations and to show the progress of the orthogonalization process.
2. By re-writing the algorithms, self-explanatory naming will be used. For example, the edge table is currently named KN (“knooppunten”). This table can be simply renamed “edges”.
3. By re-writing the algorithms, only the variables necessary for mesh operations will be used and encapsulated in appropriate classes. Class methods will use “const” qualifiers when possible, so it is clear from the function signature that internal variables will not be modified.
4. Switching to C++ enables the use of ReSharper for C++ for renaming of variables and methods and fast code browsing.

Other improvements that can be achieved by rewriting the code in C++ are:

1. By using CGAL for low-level grid operations (e.g. segment intersections, inquiring distances, directions, cross products) we can reduce the number of lines of code to maintain. CGAL has several kernels that perform operations in cartesian and spherical coordinates that can be injected as template parameters (no code duplication).
2. By switching to C++ we can use the STL (standard template library) data structures and algorithms, again reducing the number of lines of code.
3. By exploring CGAL we can discover which algorithms can be beneficial for our DFlowFM users (e.g. depth-based interpolation algorithms or constrained triangulations).
4. Abstractions will be created when necessary to promote code reusability and to reduce compilation time.

The basic mesh administration has been already rewritten to compare the performance of the C++ implementation against the Fortran implementation. Table 1 reports some preliminary results for a million-point mesh (a grid 1000 by 1000 quad cells).

**Table 1.** Fortran and C++ implementations for mesh administration routines (setnodadm + findcells) and runtimes for a million point mesh.

|  |  |  |
| --- | --- | --- |
| Implementation | Lines of code | Runtime (s) |
| Fortran | 1135 | 1.58 |
| C++ | 489 | 0.96 |



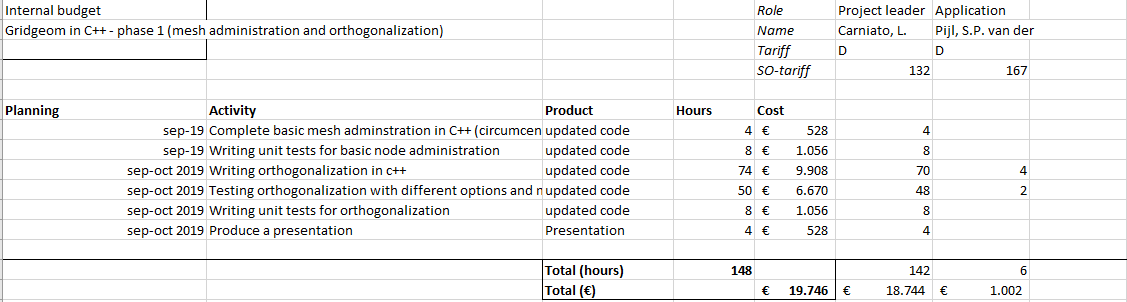
**Figure 1.** Parts of the Gridgeom library in C++. Basic variables and functions are shown on the bottom row and high-level functions are shown on the top row.

The library repository has been already configured for cross-platform compilation (linux and windows) by using CMake and can be retrieved here:

<https://svn.oss.deltares.nl/repos/delft3d/branches/research/Deltares/20190815_GridGeom>++

**Time / budget / planning / team**

See the table below, which is also available as a spreadsheet:



The requested budget for the project is € 19746,--. At the end of the project, the results with the new library will be presented to the DFlowFM kernel team. The presentation will focus on the results obtained and the C++ techniques used to obtain good performance for numerical problems.

**Risks and measures**

|  |  |
| --- | --- |
| Risk | CGAL spherical kernel can not be used. |
| Measure(s) | Spherical implementation from DFlowFM will be reimplemented. |

|  |
| --- |
| Copy for  To be defined. |
|  |