INTRODUCTION
The PyFEM open-software library is a collection of Python classes for the conversion of 2D or 3D mesh representations. It was initially developed by the Computational Geoscience group at CSIRO for use in the field of geoscientific numerical simulations, where the concept of a representative 3D mesh (consisting of cells and vertices) is often used to organise input data to a numerical solver using finite element or finite difference techniques [1]. Such computational techniques are used to model geological systems within typical phenomenological domains as mechanical deformation, fluid flow, thermal evolution and chemical reactivity [2].

UTILITY
The PyFEM mesh conversion library is designed to facilitate the use of a wide range of 2D and 3D meshed models in finite-element or finite-difference numerical modelling packages. Its primary use is to convert meshes created in one particular software package into another. This is often required in cases where:

- a numerical solver requires an input mesh data in one particular format, but the mesh is more easily created in another;
- the output from a solver is better visualised or analysed in a different mesh format or
- continuous conversion is required between mesh formats to couple one solver to another, perhaps in different phenomenological domains.

The PyFEM library (formally known as the XMLNModel library) is written in the interpretive language Python, making it relatively platform-neutral. Additional functions are provided through its associated package FEMStats, such as:

- statistical queries against a mesh or group of meshes (usually representing a time sequence),
- average sample value at a particular cell over time,
- populating a mesh with a new dependent variable,

and many more. PyFEM has been designed to be easily extensible, to encompass future or different mesh formats.

DESIGN
The PyFEM library has been developed over the last 10 years in conjunction with the Desktop Modelling Toolkit (DMT). The DMT is a toolkit that enables geoscientific modellers to graphically set up a numerical simulation in a manner that is independent of the particular numerical solver or computational resource being used [3]. PyFEM can be run via the DMT's graphical interface, or as a stand-alone library via python scripts.

PyFEM makes use of a central “abstract” mesh representation (the PyFEM class). A converter class for format X then simply provides the X -> PyFEM mapping for reading and the reverse for writing. All converter classes can take input from either a file or an open stream, allowing easy “daisy-chaining” of processes.

The primary focus of the PyFEM library is to support finite element modelling, hence the library provides representations of the following FEM concepts:

- nodes – 3D coordinate points with optional properties,
- elements (cells) – volumetric entities consisting of 3 or more vertex nodes, faces, one or more centroids, optional properties,
- regions – groups of cells (which may or may not be contiguous) that can be treated as a single “domain”,
- surfaces – contiguous groups of cell faces that define some form of boundary (eg bounding surface, stratigraphic surface),
- sampled values – properties that are measured/calculated at cells or at nodes
- boundary conditions – constraints on phenomenological behaviour at the bounding surfaces of the mesh, eg Dirichlet or Neumann type conditions for fluid flow, heat transfer etc,
- bounding box – the extents of the mesh in the global XYZ coordinate scheme.
Different finite element numerical solvers may use one or more of a variety of cell types, to optimise either topographic or numerical accuracy. PyFEM supports all of the common cell types, both linear (in the sense of a single gaussian point) or quadratic (second-order gaussians):

- Linear: triangle, quadrilateral, hexahedron, tetrahedron, wedge, pyramid
- Quadratic: triangle, quadrilateral (8 or 9 node), tetrahedron, hexahedron (20 or 27 node)

**CONVERTER CLASSES AND TYPES**

Currently the PyFEM suite supports the following formats:

- Gocad Sgrid
- Shemat
- Finley
- VTK unstructured grids (ascii or XML)
- Gmsh
- TecPlot block grids
- Patran, Abaqus Neutral format grids
- Fracis XMML4FEM grids
- Generic ASCII format grids

As an open-source development, we welcome input from developers or researchers who wish to extend the library to incorporate additional formats. Such interested parties should contact the lead author for access to the code repositories.

**SOFTWARE PROCESS**

The PyFEM library code is kept in a Subversion repository on one of the CSIRO servers. Checkins to the head branch of this repository will trigger the Buildbot system installed on the server. This will run all the unit-tests against the current version and validate the code. Once validated, an installation package for Windows and Linux is automatically created, and the installer is uploaded to a secured portal.

The unit-tests are based around a set of reference meshes that contain nodal and element properties, boundary conditions, regional information and 3D topology. They use the built-in Python unit testing framework. Each converter class and the top-level methods of the PyFEM class, have an associated unit-test. Developers are encouraged to “test first, write later” i.e. write the unit-tests first, for any new class or method added, in line with the Agile development model [4].

Branches are not automatically subjected to unit testing, but can be called manually by the developer.

**CURRENT AND FUTURE WORK**

Most of the recent work done on the PyFEM library has been to improve the computational efficiency of the conversions. Some of the FEMStats methods (eg point-in-polygram algorithms) have been recompiled in C++ and are called through a SWIG wrapper [5]. The Psyco module [6] for python provides “just-in-time” compilation of critical parts of the code for faster runtime, and is being added to various parts of the library. The alternative pure-python functions are still available.

There is on-going work in adding new converter classes in order to support new mesh generation software.

**REFERENCES**

5. A tool for wrapping C/C++ functions and calling via scripting languages such as Python. See [www.swig.org](http://www.swig.org).
6. See psyco.sourceforge.net
ABOUT THE AUTHOR(s)

Dr Gordon W.H. German received his PhD in Remote Sensing and Artificial Intelligence in 1998. He joined CSIRO as a computer scientist in 1999. During this time he has been involved in various projects surrounding geoscientific numerical modeling, workflow analysis, grid computing and high performance computing. He is currently working in the Technical Algorithms team, Computational Geoscience, of the CSIRO Minerals Down Under flagship.

Since joining CSIRO in 2003, Dr Thomas Poulet has been working on various projects aiming at transforming mineral deposits' formation understanding from a qualitative to a more quantitative and predictive science. His activities have covered numerical modelling, signal processing, mathematical optimisation and visualisation of high dimensional model parameters, software engineering, e-Research, grid services and high performance computing. His current research activities centre on reactive transport in porous media and thermodynamics of dissipative processes, which was the subject of his PhD. Dr Poulet is also in the Technical Algorithms team, Computational Geoscience, of the CSIRO Minerals Down Under flagship.