LLNL-TR-408094

Report of the IMFIT Advisory Committee from the Meeting of 9/16/08 at General Atomics

R.H. Cohen, J. Cary, W. Houlberg, D. McCune

October 23, 2008

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Report of the IMFIT Advisory Committee from the meeting of 9/16/08 at General Atomics

Committee: Ronald Cohen (chair), John Cary, Wayne Houlberg, Douglas McCune

The Committee commends and thanks the presenters and GA for providing clear and informative presentations. We present below our responses to the questions raised in the Charge, as well as additional comments that may be useful for the project.

1. The IMFIT framework is based on PYTHON and CCA; can IMFIT meet its long-term goals with this choice of language for framework?

Our first recommendation is to change the terminology; it appears that the project is not really utilizing CCA (Common Component Architecture). The project is really about using Python to organize workflow among components that are spawned as processes via pipes from Python. We suggest that in describing the project, the team drop the designation "CCA" and simply state that the IMFIT framework is based on using Python to control workflow among components.

Concerning the specific question: we understand the principal goal of the project to be the development of an easy-to-use tool that efficiently integrates different physics modules to support experimental data analysis and modeling. Yes, Python and the workflow-based framework are appropriate choices for meeting this objective.

The Committee, in examining the goal, discussed whether the IMFIT project was duplicative of other projects; there are several others in the U.S. and elsewhere with integrated modeling goals. In particular in the U.S., SWIM, PTRANSP, and the proto-FSP's. However this project occupies a reasonably unique niche through its emphases on integrating experimental data, use of existing components, its near-term focus, and the extent to which is targeted at experimentalist users.

2. Will IMFIT benefit from the additional use of an interoperability language for framework such as BABEL?

No, IMFIT is not really doing inter-language communication. In particular this will be the case as long as inter-component communication is via files -- and that's OK so long as you aren't doing lots of short steps with communication at each step.

3. IMFIT GUI is based on the public PyGTK toolkit; will IMFIT benefit from the additional use of other public Python graphic toolkits such as wxPython that is a cross-platform wrapper, or PyQt, or a commercial package such as IDL ?

PyGTK seems adequate for now. The team needs to decide if it needs more capability. Avoid licensed software that interferes with portability and the open-source goal (and thus in particular avoid IDL).

4. Will IMFIT benefit from using a XML-RPC as alternative to sending files thru sockets?

IMFIT could benefit by adopting some kind of self-describing format, such as XML or HDF5, but data can still be sent over sockets. This is part of a broader question: "Should IMFIT adopt some kind of standard for communicating data between components?"   To do anything other than what is done now requires modification of individual components (or convertors/wrappers). But moving to a self-describing standard is a good idea, which should be considered. You don't need to impose a standard, but you could decide on one and move toward it gradually, as the architecture doesn't impose a standard. But whether you adopt a standard and the pace of moving toward it is up to you.

5. Will IMFIT benefit from any other available computational tool?

(a) IMFIT would benefit from fuller use of an already incorporated tool -- python language in the task execution specification, to enable desired features like branching.
(b) If there is a high degree of data hierarchy, IMFIT should explore HDF5 as well as NETCDF and compare.
(c) IMFIT would benefit from moving toward common data standards
(d) It would be useful for the IMFIT team to monitor developments in related projects such as the proto-FSP's and PTRANSP to ascertain if there are potentially useful tools.

Other questions/suggestions:

We still don't understand why there are three separate functions (init, step, final), if all three are always called together and only once in execution of a task. However, one could make genuine use of a repeated "step" as part of a python script-driven procedure.

It seems that there are two separate functions for task files – specification of data and control of the execution of tasks. Right now execution of tasks is all based on dependency analysis; one could shift to procedural scripts based on Python. This would make it easy to incorporate conditional branching and other control features that the team notes is desirable. One could adopt a hybrid approach with high-level script that exploits the existing IMFIT dependency analysis.

It was not clear to some of us that there is an advantage (in terms of simplicity) to the data specification format used in IMFIT versus direct assignment via the Python language.

Error handling needs to be better addressed-- perhaps by components passing their error flags/messages through their interface, and then using Python's error handling to centralize reporting of error messages.

IMFIT needs a reset method: if a step fails, one needs to be able to back up and do something else (e.g. change the timestep, change a code setting, and then retake the step); for long running, IMFIT will likely need a capability for state capture/saving.

Security issues: for local use, there isn't much of an issue. To seriously address security is probably too burdensome a task for the available staff.

It would be good to implement journaling/streaming of log files. It is desirable to be able to save the history of a simulation. Provenance should include version numbers of all components.

This report was prepared under the auspices of the U.S. Department of Energy, in part by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA2734.