The FACETS project: an approach to multiscale modeling by an interdisciplinary team

JR Cary, Tech-X, for the FACETS team

The FACETS (Framework Application for Core-Edge Transport Simulations) project had the goal of building a flexible and extensible, yet performant computational application for whole-device modeling. (FACETS was proposed in 2006, funded in Jan 2007, and concluded at the beginning of 2013.) FACETS had to integrate components developed by many different teams with very different philosophies (which it needed to standardize/inter-adapt to make any progress), it needed to address software issues of robustness and usability, and it needed to address data analysis and visualization. Further, FACETS had the goal of being able to take advantage of large-scale parallelism. In addition to those preconditions, FACETS needed, going forward, to integrate a team with wildly different backgrounds in the target application area, fusion plasma physics, as well as in applied mathematics and computer science. To put a finer point on this, a significant fraction of the FACETS team might have heard of gyrokinetics before joining the project but could likely not provide the basics to making such an approximation, while a different fraction (minimally overlapping) might have heard of a linked list but probably had never used one in practice.

1. Approach to executable development

The FACETS approach was to make a single executable (MPI for distributed memory computing but otherwise one memory space) that would mirror the evolution of the plasma, its interaction with its environment, and its generation of raw diagnostic data. This approach allowed us to address the mission of taking advantage of massively parallel, distributed-memory hardware. At the same time, it was recognized that many other applications would have to be invoked for problem setup and data analysis and visualization (discussed later).

FACETS approached this problem with what Tech-X personnel would now call our standard computational layering. This layering identifies the dependencies at the coarsest application level. Adherence to layering provides separation of concerns and allows component based testing, which facilitates both problem discovery and solution. The realization of the layering is that different parts of the software are put into different directories, the source code of a directory is built into a library, and the libraries are combined to create the final executable. The Tech-X standard computational layering further separates this into utilities, data structures and interactions, implementations, and integration.

Further progress requires problem analysis. For FACETS this largely followed what most fusion scientists would consider fairly standard: a core region dominated by one-dimensional transport, and edge required to be 2D (or more), and a interaction with the material walls, which can charge with and discharge the plasma species (deuterium and impurity species).

However, FACETS did not follow the standard of "a 1D transport code backbone". Instead, top-level components had their own independent lifetime and evolution and were ready to progress as soon as needed information was received, then idle upon providing the desired results at the conclusion of its calculation. This targeted allowing components to exist across core-edge regions. For example, RF sources affect both core and edge dynamics. Having RF or equilibrium "belong to the core" would prevent a simulation with more fidelity in both areas. In contrast, local transport coefficient modules (e.g., GLF23, TGLF, embedded turbulence) can "belong to the core", as they are used by only the core.

FACETS accomplished its goals. Within the five-year period we provided a modern, object-oriented framework. We did core-edge simulations, with our core solver making use of community transport modules. A visualization of a core-edge simulation is shown in Fig. 1. FACETS provided algorithmic advances as well. We built upon the Newton approaches to core solvers, adding a V-cycle approach to improved robustness. Unfortunately, these advances were not brought to publication, as they were finishing at about the time of FACETS termination.

2. Workflow issues

Here, workflow is defined as the orchestration of multiple applications to provide a scientific or engineering result. It is typically one-way (a Directed Acyclic Graph) but fits within a larger, iterative process that is common to scientific discovery and device design. As noted above, the FACETS view was that the workflow enabled problem setup as well as the further analysis of the raw diagnostic data.

Workflow addresses the problem outside of the prediction of plasma dynamics, but it is critical. Just as a plasma experiment gives one nothing without programming of controls, including coil currents, and analysis of diagnostic data, neither does a computational physics application provide much knowledge without the ability to set a problem up correctly, both initial and boundary conditions,

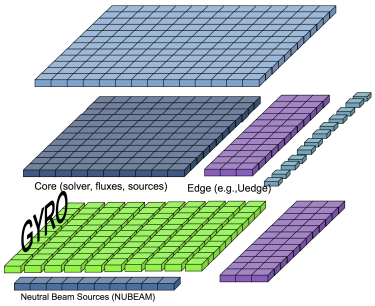

Fig. 1. Component layout per processor. Components have their own allocation of processors, with the core, edge and wall components stood up as independent (MPI) processes. The framework set up the mediation between these first-rank components. Core calls out to components, such as transport coefficient modules (GLF23, GYRO, etc), that only it needs. By the end of the FACETS project, neutral beam sources were still a core owned component, but long term they would have been migrated to an independent component that could live across core and edge.

The most basic element of a workflow is the currency, the means by which data is stored and organized. This is partly the format. [TRANSP uses MDS Plus and NetCDF, UEDGE at the outset used PDB (an LLNL home-grown product).] FACETS needed a format suited for parallel computing. The second issue is how the data is stored in the file. Ideally, that method should allow for data discovery; metadata should allow a reader to discover the semantics. Ultimately, FACETS developed an open standard, VizSchema, which defined the organization and mark up data within an HDF5 file, defining also the basic semantics associated with visualizable items. All components used by FACETS were modified to provide output in VizSchema. A VizSchema plugin was developed for the VisIt application; it is also available within ParaView.

For workflow control, FACETS developed python scripts. For example, there is a need to acquire data from experiments for initial profiles and for equilibrium, and from the latter to get geometric data, which is used for the profile evolution and for edge evolution. Final data from the core evolution is 1D data, which for the visualization shown in Fig. 3 has to be put on the equilibrium mesh in a VizSchema compliant manner.

A follow-on proposal envisioned a system in which IPS managed the workflow, while FACETS provided the plasma dynamics simulation. The marriage of these two technologies was started but not consummated due to termination of both projects. FACETS did not engage any of the visual programming approaches to workflow management. This follows developments at large, where the one has seen the decline of OpenDX (a visual programming approach to visualization) with replacement by VisIt and ParaView, which through a GUI allow one to set up pipelines from data to visualization.

FACETS did foray into providing some simple GUI capabilities. Since then, we have developed a more sophisticated approach to setup. Making codes easy to use is not highly valued within the scientific modeling community, where users are mostly developers, but for design engineers it is paramount. Hiring an engineer at a design firm and having that person take a year or more to become productive is not even considered. Such long times to productivity may be okay at DOE labs, but only if said person will be in place much longer than in training. This appears to be an area for further evaluation by FES/ASCR, if broadly usable codes are envisioned.

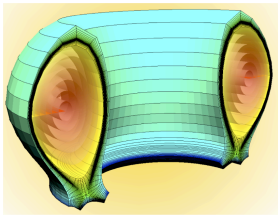

Fig. 2. Visualization of a core-edge simulation. This simulation used the FACETS core solver with UEDGE for the edge model. The transport coefficients in UEDGE came from an interpretive analysis. This visualization is a rotation of the actual system, which was axisymmetric. This visualization was critical to assuring that we had the grids from the core and edge components matched up.

3. Curation

The FACETS infrastructure as it particularly applies to fusion is curated in a subversion repo, https://ice.txcorp.com/svnrepos/code/facets/trunk. This is used primarily as an external to https://ice.txcorp.com/svnrepos/code/facetsall/trunk, the facetsall repo, which external'd in many modules that FACETS developed and relied upon, such as facetsifc, which defined the basic interfaces, and fciowrappers, which provided an implementation neutral layer for I/O to either NetCDF or HDF5.

Inevitably, bit rot sets in. We are assessing its degree with this initiative. We have determined that the repos are in place with all of the externals. (The Bilder and SciMake repos were moved to SourceForge, but that happened in June and November of 2012, just prior to project end.) However, some builds currently fail on OS X due to compiler changes.