Project Name

The Eclipse Integrated Computational Environment

Jay Jay Billings, ORNL
20140219

Parent Project

None selected yet.

Background

The science and engineering community relies heavily on modeling and simulation to bridge the gap between theory and experiment. Almost every type of physical phenomena has a corresponding community of codes that simulate it for various physical conditions and parameters. A wide array of modeling tools exist to create inputs for these simulators. In fact, modeling and simulation tools are so widely available that the challenge is not finding one that will simulate the system, but actually *figuring out how to use it.*

There are four primary activities performed by all computational scientists and engineers: setting up the model, launching the job, analysing the results and managing the input and output data. There are certainly other activities, such as building or authoring code, but these “Big Four” are the only ones that are shared by users and developers. These activities may be trivial for a super user of any one code or set of codes, but new users can be severely challenged and even put off by these tasks. In some cases the challenges are scientific in nature or related to the engineering, but, arguably, in the majority of cases the challenges come from details like file formats, exotic host platforms and poor or completely missing usability.

The details of the way the Big Four activities are performed for any subset of the community are certainly different, but conceptually there are remarkable similarities because of shared packages, ideas and data. While it would be impossible for any one tool to service the entire community, ideally, this commonality could be exploited to create a suite of integrated tools to support users. This integrated tool suite would be a platform to which developers could contribute their own tools to help users and one that users could leverage to accelerate their modeling and simulation studies with multiple tool sets and techniques. The details of exactly how files are manipulated and stored and jobs are launched and monitored would be encapsulated, automated - where possible - and otherwise unexposed to the user.

We propose a new Eclipse project - the Eclipse Integrated Computational Environment - that leverages the vast collection of Eclipse tools in the Integrated *Development* Environments to develop this integrated tool suite. The existing Eclipse tools - advanced editors, user interface components, parallel tools, etc. - will be used to develop science and engineering tools focused

on modeling and simulation. New tools will also be developed and contributed that add capability to the Eclipse Platform for science and engineering tasks that have not, to date, been added to the Platform.

Scope

The scope of this project will cover everything necessary for performing the Big Four activities in addition to streamlining some of the activities necessary for obtaining modeling and simulation codes. This includes:

- Input generation using existing or new I/O tools for modeling and simulation projects.
- 3D geometry construction for physical models.
- 1D, 2D and 3D meshing tools or connectors to existing third party tools.
- Database connectors for scientific and engineering databases for physical information.
- Tools for launching scientific and engineering codes on local or remote machines and managing the input and output of those jobs.
- Tools for monitoring remote jobs with domain-specific views.
- Scientific and engineering tools for viewing data, including but not limited to plotting routines, 3D visualization, scientific data mining and “low-fidelity, high-access” views for derived quantities for modeling and simulation.

In all cases, an emphasis will be placed on creating reusable tools that can be easily deployed for multiple modeling and simulation projects instead of creating highly specialized tools for a single modeling and simulation project. The first case maximizes the benefit of the tool suite to customers and accomplishes the second, with proper design, but the second case does not accomplish the goals of the project.

Description

The Eclipse Integrated Computational Environment (ICE) will address the usability needs of the scientific and engineering community for the Big Four modeling and simulation activities. The focus of the ICE will be to develop an easily extended and reusable set of tools that can be used by developers to create rich user interfaces for their modeling and simulation products. Custom widgets and data structures with well-defined interfaces and high-coverage unit tests will be provided for plugin developers. Plugins based on the ICE tools will also be developed and released with the ICE for codes that the developers use and the community contributes to the extent that resources allow. The idea is that the tools should be available for developers to do what they do and the deployment mechanism should be ready and waiting when they are finished.

Each of the big four tasks presents unique challenges, but much of the capability required can be handled by extensions to the existing Eclipse tools.

Creating models for scientific and engineering projects is a very difficult task that is, presently,

very time consuming and hampered by the lack of an integrated tool chain. For example, consider the screenshot in Figure 1 of an input for a battery simulation configured using the code in the Initial Contribution.

The activity depicted in Figure 1 appears to be straightforward. Some information is required to pick a case and some other information is required in a table. After that information is provided, some buttons can be clicked and an input file will be generated. This seems relatively simple when described as such and presented with an Eclipse Form, but on the backend it requires detailed knowledge about the input format and four simulation codes. It also requires dependency tracking and a little bit of math. Both the input format and the four simulation codes are relatively well known in the battery simulation community, but no tools exist for using them easily.

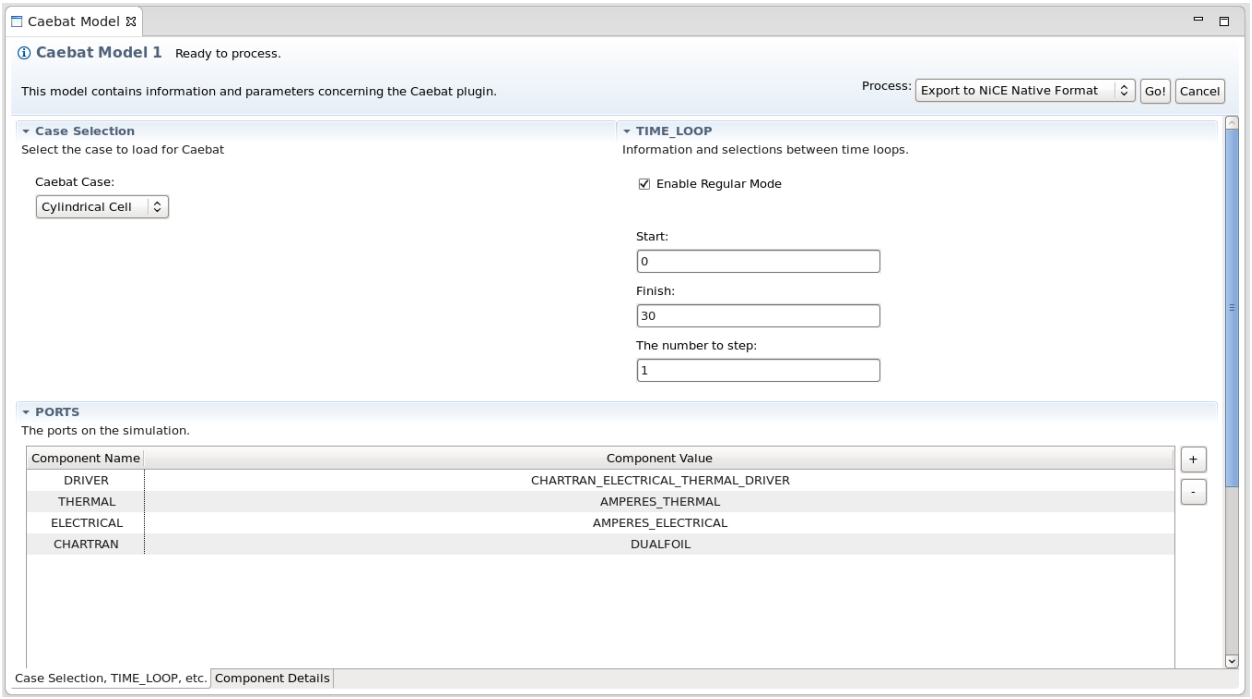

Figure 1 - Setting up the input for a battery simulation.

A user who was not using the tool would have to learn all of those things themselves, much like a Java programmer not using Eclipse for Java Developers or a similar tool would have to learn the details of using the “javac” and “java” command line tools. However, with the tool the user can perform that task far quicker.

The Eclipse Form in the example takes advantage of several data structures and widgets that are automatically created and rendered based on very little information from the developer. The instructions used to draw the widgets on the screen are separate from the domain-specific information (business logic) provided by the plugin developers. While the developer is responsible for saying that they want information from users about time integration or component

configurations, they *never* have to be responsible for drawing that information. An extension API exists, but they can (and tend to) leave that task up to the computer scientists.

The second of the Big Four activities, launching jobs, is in some ways the defining activity of modeling and simulation. It can also be extremely difficult if the simulation in question requires an exotic machine, a special operating system or really any system with which the user is not intimately familiar. In interviews, users tend to say that they would rather push a button, enter a password if the job is remote and watch the information stream back than “work the shell” themselves. In some cases a user may not even know how to use a shell on the operating system in question and cannot easily run the job.

Similar to the proposed modeling tools, the ICE will provide a simple developer interface for configuring job information - by template - for users. These tools will handle input matching with jobs, the actual execution and streaming information back to the user in an organized and recoverable way that is easily reproducible. This burden on users and developers is simplified to that of saying *what* needs to be launched, although one does this in the user interface and the other writes, for example, a reusable, templated shell script.

Analyzing and managing data, the last two of the Big Four activities, go hand in hand. There are some cases where data can not be managed or analyzed on a local machine because the data files are too big. The exact ways of doing that with a suite like the proposed ICE are open areas of research. However, for smaller data sets there are many things that can be done to improve usability.

The most obvious thing that can be done is to provide users a way to read their data. It is increasingly common in the science and engineering community to use a binary data format, so providing some way to read that data is key. The second most obvious thing to do is provide simple information on the provenance and present location of the data. (That may seem obvious, but it is commonly skipped in scientific codes.)

It is much more useful to provide visualization capabilities for users that can be pre-programmed by developers that help them extract knowledge from the data, as shown in Figure 2. It is important to make a distinction between different types of data since there are different types of users. Some users will prefer full 3D views of their simulations on the mesh as shown in the left pane of Figure 2. Other users will prefer parameterized views of derived macroscopic quantities as shown in the right pane of Figure 2.

In both cases it is important to make these tools easily accessible to users and plugin developers. The view on the left of Figure 2 uses an extension that enables viewing 3D meshes in Eclipse with VisIt, https://wci.llnl.gov/codes/visit/, which some years ago was successful in rendering images with over one trillion elements of data. The parameterized view in the right pane of Figure 2 is also a good example. It is a cartesian grid drawn with the Graphical Editing Framework and is fully interactive, resizable and can be exported to an image for users with a

few clicks. It has a simple interface for user interface developers and can handle hexagonal and circular grids in addition to cartesian grids.

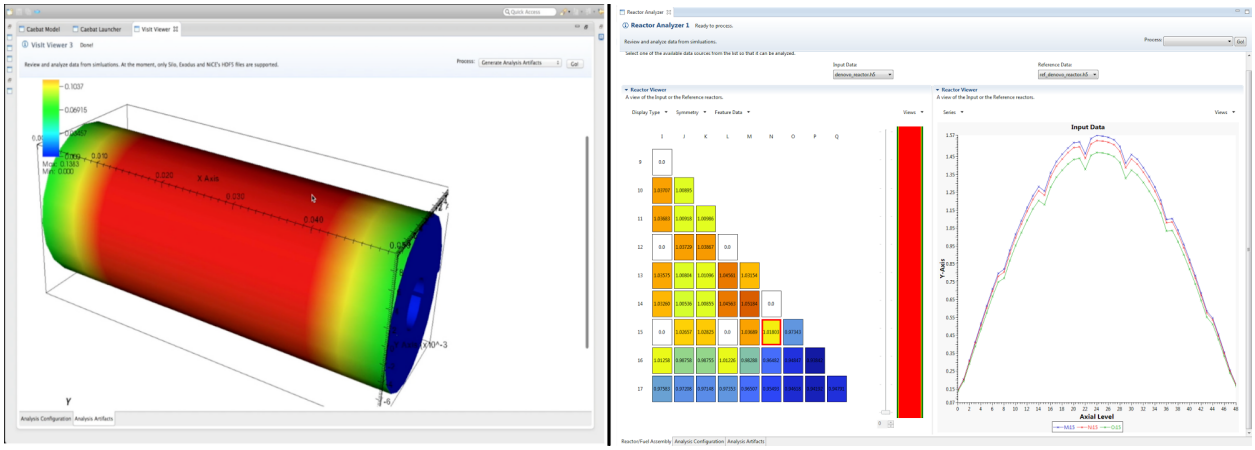

Figure 2 - A 3D image of the mesh in a battery simulation on the left and a parameterized view of a nuclear fuel assembly and the power on a few rods.

Why Here?

This project is a natural fit for the Eclipse Foundation for several reasons. The most important reason is that the scientific and engineering communities are starting to use Eclipse more. The formation of the Eclipse Scientific Industry Working Group is a sign of things to come, namely that more scientific and engineering entities are relying on Eclipse for their tooling infrastructure or at least showing interest in it. A corollary to this is that these capabilities do not already exist in the Eclipse Platform even though they are well represented in the broader Eclipse ecosystem.

The Eclipse Platform is also, in the opinion of the authors, the only environment that already has a vast majority of the capabilities needed to produce the integrated tools suite. Tools that are used for authoring can be reused very easily for modeling and simulation. The platform has over ten years of reusable capability that could not be easily or cost effectively reproduced by any single entity.

The Eclipse Foundation also offers an avenue by which contributions to the integrated environment can be tracked, vetted and released more broadly and easily than alternatives. This minimizes risk and encourages adoption by corporate customers, which is something that helps transfer technology from Academic and National Laboratory circles to industry.

Legal Issues

There are no obvious legal issues with the code. It includes and depends on two open-source

codes, the Hierarchical Data Format version 5 (HDF5) and JMonkeyEngine version 3 (JME3). Both are released under either Original or New BSD licenses.

Initial Contribution

The Initial Contribution (IC) will be made by UT-Battelle, LLC, which manages the Oak Ridge National Laboratory (ORNL). The IC will be from an existing project, the NEAMS Integrated Computational Environment, presently released by ORNL at Sourceforge.net.

The initial contribution is an Eclipse RCP-based product and relies heavily on other Eclipse projects. It relies especially on the Parallel Tools Platform, Plugin Development Environment, the Graphical Editing Framework and Tycho.

Project Scheduling

The initial contribution will be made in quarter two of 2014 and the first release would happen by the end of quarter one of 2015.

Future Work

Future work for this project in the short term will include improvement of the core tool set, data structures, widgets and adapters for third parties and plugin developers. It will also include the development of plugins for “high priority” modeling and simulation codes to both the project members and community.

Long term work on this project will focus on data sharing and mining and team collaboration activities. This includes tools for automatically mining modeling and simulation data as well as tools for easily sharing that data, or its metadata, in an IP-friendly way between collaborators. Improvements in the visualization capabilities will also be critical to long term success.

Interested Parties

The initial contribution to this project will come from an existing project on Sourceforge.net and an on-going development effort within the U.S. Department of Energy to develop a computational environment for modeling and simulation. Both the Advanced Modeling and Simulation Office of the Office of Nuclear Energy (NEAMS) and the Office of Energy Efficiency and Renewable Energy (EERE) will maintain a long-term interest in the project as the primary funding sources for the developers.

The Oak Ridge National Laboratory has several projects that use the existing product and will maintain an interest in the project for the foreseeable future.