Component-Based Application Code Development, Part 1: The Agile Components Strategy and Albany Code

Andy Salinger
Sandia National Laboratories
Irina Tezaur [SNL], Mauro Perego [SNL], and the Albany and Trilinos development teams

Abstract

In this submission, we present a code development strategy and collaboration model that is designed to maximize the leverage and long-term impact of algorithmic investments on application code projects such as earth system models.

Our computational science organization has been following a strategy -- dubbed "Agile Components" -- for code development that puts the emphasis on generating and leveraging reusable software components [1]. Software components are broadly defines as libraries, interfaces, software quality tools, and demonstration applications. We have found there to be numerous benefits of this approach in the writing of new application codes in terms of development time, costs, and algorithmic capabilities. Furthermore, we have found that the Agile Components approach provides a framework where scientists and algorithm experts can collaborate on a code project and each can be productive and successful in their own realm, and leveraging of expertise in funding is maximized. In this way, it maps well to the vision and funding mechanism of DOEs SciDAC program.

In this submission (Part I), the Agile Components strategy and collaboration model is presented, along with its manifestation in the Albany code framework. In Part II (submitted by Irina Tezaur), the experiences of using the Agile Components approach in the development of the Albany/FELIX Ice Sheet application code under the PISCEES SciDAC project will be presented. We believe this example is instructional for creating a model for future successful Climate Science – Computational Science collaborations.

Background: The Agile Components Strategy

The Agile Components strategy for computational science organizations involves requiring projects to **both leverage and contribute to** a comprehensive set of software components, consisting of libraries, interfaces, software quality tools, and demonstration applications.
Just as compilers, BLAS, Lapack, and MPI have long been standard external dependencies for application codes -- and it is also common to depend on external linear solvers and meshing tools – the Agile Components strategy extrapolates this trend to include dozens of algorithmic capabilities that can be generalized into reusable libraries. This includes such diverse capabilities as automatic differentiation, discretization tools, and mesh databases. The strategy also extends to a common and mature set of software engineering tools, so that each application project doesn't have to become experts in this arena as well. Figure 1 conveys the large amount of leverage from other development efforts that align with the development of the Albany/FELIX ice sheet code.

Many benefits and challenges to this strategy have been enumerated in a technical report [1]. Benefits include the ability to leverage several funding sources to establish an algorithmic expertise, the ability to leverage that existing expertise on new applications, the amortization of verification, porting, and maintenance costs across projects, and the modular design of codes to allow for project agility. The main drawback is the dependence on code written by others.

Research To Date: Albany and Trilinos

At Sandia, we have been actively pursuing and refining this strategy for a number of years. The common set of reusable libraries and interfaces is deployed primarily through the Trilinos suite of computational math algorithms [2]. The Albany code project [3] is the chief incarnation of the strategy. Albany is a finite element code base that hosts several funded application projects, is the testbed for several algorithm research projects, and provides a template for a science applications code's software engineering tools and processes.

In Albany, we have successfully developed application codes for quantum device modeling [4], computational mechanics [5], and an ice sheet velocity solver [6], and are making progress on an atmosphere dycore. These codes are all born with a quality software engineering infrastructure, scalable algorithms, and embedded analysis and UQ algorithms, and are on a path to performance portability across all architectures. For many of those involved, the productivity in developing new applications with sophisticated capabilities in Albany using Trilinos libraries has been striking. While this assertion is anecdotal, not easily quantifiable, the success of these projects is supporting evidence, as is the fact that Agile Components is central to Sandia's execution strategy for its ATDM effort, a significant new code development effort under NNSA's ASC program.

In the companion Part II idea paper submission by Irina Tezaur, the experience of using this strategy for the development of the Albany/FELIX Ice Sheet code [6] will be detailed.

Proposed Direction: Agile Components for Climate Science Impact

The strategy described here is designed to maximize the long-term productivity in developing coupled earth system models, in terms of the speed and cost of development, the breadth and sophistication of the algorithmic capabilities, and the trustedness of the simulation results.

With those goals in mind, the path forward for impacting Climate Science **requires the climate application codes to embrace the use of software components developed by computational scientists and in developing long-term collaborations**. The benefits are numerous, including:

1. Inclusion of the algorithmic capability into the climate code.
2. Access to the algorithmic expert for continued engagement, who can work in his/her own familiar code base. From this foundation, fractional funding streams and even consultations can have large impact.
3. Automatic, free upgrades in capability as the library development continues through other funding sources and using developments motivated by other applications.
4. A decreased code base that must be supported, and ported to new architectures.
5. Encourages more modular code design, for improved agility and maintainability.
6. Broadening of the network of people thinking about your application.
7. Ability to leverage algorithm and inter-disciplinary funding streams to impact climate science application development.

In contrast, an alternate collaboration model is for the application scientist to learn about a promising algorithm and implement it directly in the application code. This shares benefit #1 above, and has the added benefit of the application scientist having complete control and familiarity of the code base, but sacrifices the remaining benefits from the list above.

A separate phase of the strategy can involve spinning-off algorithmic capabilities that were originally part of climate applications into reusable libraries to be deployed for other applications. This has many benefits, including the expansion of impact of the initial investment, the maturation of the algorithms in other settings, the access to other funding sources to continue the work in a more general setting, and career growth for the developer.

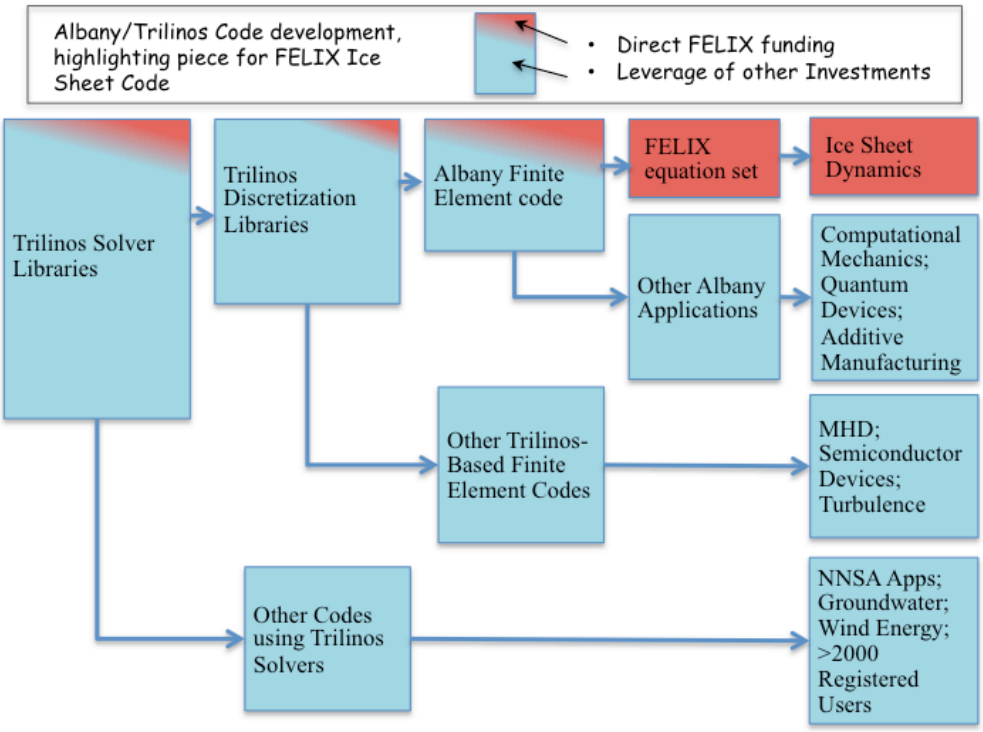

Figure 1: A schematic showing the leverage of the Agile Components code strategy, highlighting the slice of the code base written specifically for the Albany/FELIX ice sheet application. It can be seen how large parts of the code base are also used by other applications -- several of which are listed in the right column -- which all contribute to the development costs, research drivers, and creation of expertise.

References

1. A.G. Salinger, "Component-based scientific application development," SAND2012-9339, 2012.
2. M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, A. Williams, and K. Stanley, "An Overview of the Trilinos Project," *ACM TOMS*, 31(3), 397-423, 2005.
3. A.G. Salinger, R.A. Bartlett, Q. Chen, X. Gao, G.A. Hansen, I. Kalashnikova, A. Mota, R.P. Muller, E. Nielsen, J.T. Ostien, R.P. Pawlowski, E.T. Phipps and W. Sun, "Albany: A Component-Based Partial Differential Equation Code Built on Trilinos", Sandia National Laboratories Tech Report SAND2013-8430J, 2013.
4. X. Gao, E. Nielsen, R. P. Muller, R.W. Young, A. G. Salinger, M. S. Carroll, "The QCAD Framework for Quantum Device Modeling," *15th international Workshop on Computational Electronics*, 1-4, 2012.
5. W. Sun, J.T. Ostien, A.G. Salinger, "A stabilized assumed deformation gradient finite element formulation for strongly coupled poromechanical simulations at finite strain," *International Journal for Numerical and Analytical Methods in Geomechanics*, 37(16), 2755-2788, 2013.
6. I. Tezaur, M. Perego, A. G. Salinger. R.S. Tuminaro, and S.F. Price, "Albany/FELIX: a parallel, scalable and robust, finite element, first-order Stokes approximation ice sheet solver built for advanced analysis," Geosci. Model Dev., 8, 1197-1220, 2015.