SYNERGIA: AN ADVANCED OBJECT-ORIENTED FRAMEWORK FOR BEAM DYNAMICS SIMULATION

D. Dechow, P. Stoltz, Tech-X, Boulder, CO 80303, U.S.A
J. Amundson, P. Spentzouris, Fermilab, Batavia, IL 60439, U.S.A.

Abstract

Synergia is a 3-D, parallel, particle-in-cell beam dynamics simulation toolkit. At the heart of the software development effort is the integration of two extant object-oriented accelerator modeling libraries—IMPACT written in Fortran 90 and mxyzptlk written in C++--so that they be steered by a third, more flexible human interface framework, written in Python. Recent efforts are focused on the refactoring of the IMPAT-Fortran 90 codes in order to expose more loosely coupled interfaces to the Python interface framework.

OVERVIEW

The Synergia beam simulation framework is comprised of a wide variety of software libraries and toolkits. Like any non-trivial software solution, there is a significant ongoing software development effort associated with the Synergia project. The overriding software engineering goal is to take the existing software pieces and create a more flexible and extensible whole. The resulting software architecture then will be more amenable to incorporating new functionality, in the form of new physics modules, new methods for extracting meaningful knowledge from simulations, and new components that make developing simulations more intuitive and effective. This paper describes the ongoing effort to create a more flexible and extensible architecture.

Currently, the physics simulation routines reside in the IMPACT [1] and mxyzptlk/beamline [2] libraries. The IMPACT library provides a parallel implementation of particle propagation, RF modeling, and parallel space-charge calculations. The mxyzptlk/beamline libraries provide support for a broad base of accelerator simulation and modeling. In the Synergia framework, the mxyzptlk/beamline libraries are used primarily to generate transfer maps from MAD language descriptions.

Additional frameworks and toolkits are used for visualization, parallelization, and data analysis. Some of these toolkits (only those whose inclusion had an effect on the overall direction of the Synergia framework) are described in later sections. Figure 1 gives a graphical representation of the major components that are present in the architecture of the Synergia project.

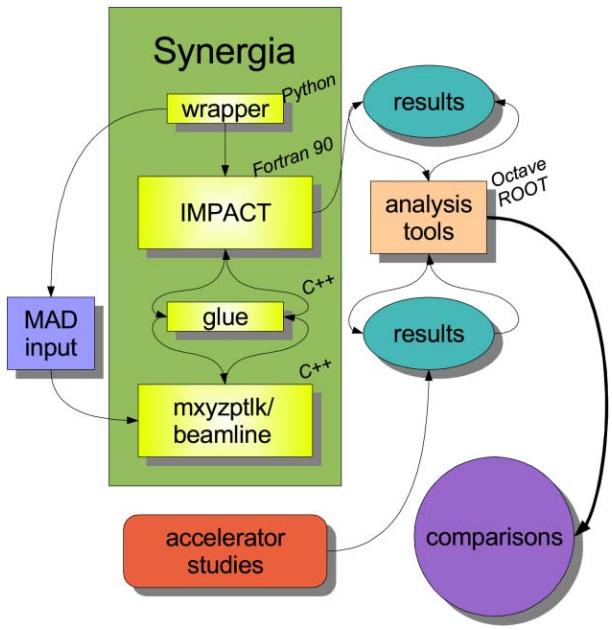

Figure 1: Synergia components.

SYNERGIA SOFTWARE DEVELOPMENT

A major design decision which has had a downstream effect on the entire architecture was the choice to make Synergia a Python-steered framework. This decision to drive the Synergia framework via Python was made in order to provide a more humane user interface. Previous versions of Synergia depended on a tedious, error-prone, file-based solution in order to populate simulations with parameters.

By making Python scripts and the Python interpreter the primary mode of user interaction with Synergia, we are able to envision a software system in which every other component in the system disappears into a Pythonic background.

An initial effort which took place last year was devoted to placing the main loop of IMPACT under the control of Python. This initial, exploratory development was only possible because of the existence of the Forthon [3] wrapper generator tool and its ability to work with Fortran 90 modules. As a secondary benefit, the Synergia project team members were able to gain valuable knowledge concerning the internal structure of the IMPACT library. The Synergia architecture that resulted from this exercise can be seen in Figure 2.

This experience also demonstrated that simply wrapping the main simulation loop of IMPACT was not going to provide the type of flexibility that was deemed necessary for adding new accelerator physics. To this end,

0-7803-8859-3/05/$20.00 ©2005 IEEE

it was decided to undertake an exercise in application programming interface definition via prototyping.

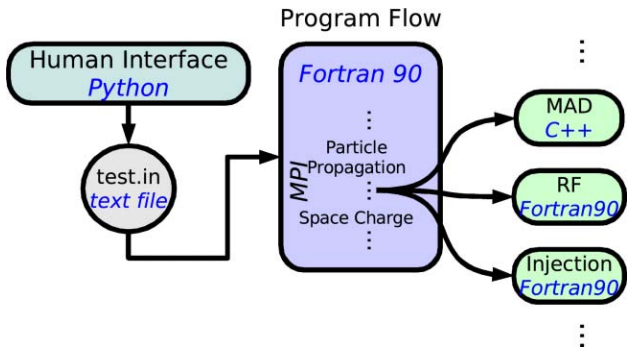

Figure 2: Old Synergia architecture.

Prototyping Efforts

The most recent prototyping exercise has been guided by the goal of being able to define the minimal set of IMPACT components that is necessary to develop a meaningful Python-steered simulation. After identifying the appropriate components, we can experiment with their interfaces by wrapping them with Forthon. Once the components are loaded into a Python interpreter, we then can use Python's extensive runtime capabilities to further experiment with the components.

The development process allows us to separate and tease apart the necessary components from the rest of the library. The process again has been invaluable in terms of what we have learned about the internals of IMPACT. It has also demonstrated sufficient justification for undertaking a more formal refactoring effort.

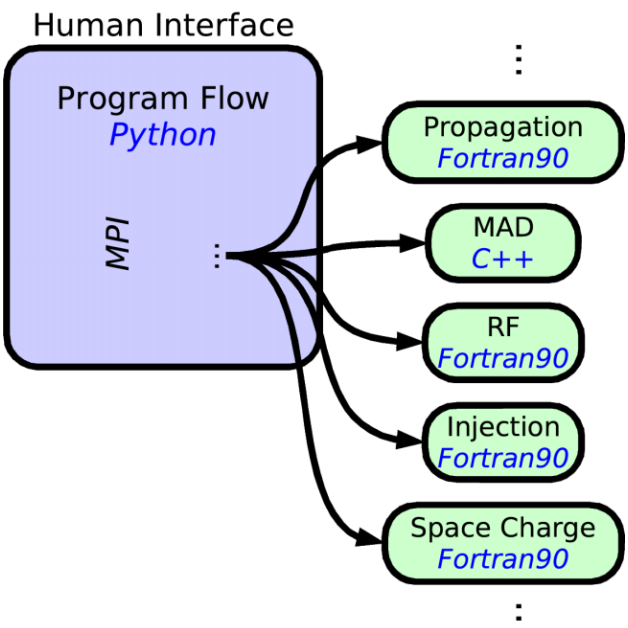

Figure 3: New Synergia architecture.

Future Refactoring

Refactoring an existing software architecture is a set of processes for modifying the structure of a software system while preserving its original function [4]. It is generally understood that refactoring techniques are used to facilitate the evolution of a software architecture.

In order to verify that the system's behaviors have been preserved, it is necessary to develop a thorough testing suite. This is an ongoing effort, and it has been aided by our other investigations into the structure and function of IMPACT.

A standard set of Fortran refactorings, such as cleaning up the global namespace and adding parameters to subroutines is already underway. A more rigorous, structure-based refactoring will be performed once the final set of internal software interfaces has been defined.

The ultimate goal of the Synergia prototyping and refactoring efforts is a software architecture wherein all of the system's components have a well-defined Python interface (see Fig. 3).

APPLICATIONS

The intended outcome of the prototyping and refactoring processes is a more flexible and extensible software system. These qualities will facilitate the adding of new physics modules. Among the proposed additions are beam-beam, impedance, and possibly electron cooling modules. The beam-beam effort is briefly described in the next section.

We intend to extend Synergia by incorporating a module that will be useful in understanding the characteristics of high-intensity colliders. At higher intensities, beam-beam collective effects become increasingly important. This module will be an extension of the parallel calculation of these effects as is described in Qiang [5].

TOOLS

In addition to the libraries and software tools that have been described above, the Synergia framework takes full advantage of the glue-language capabilities that Python offers by integrating a variety of other tools.

Octapy

Octapy is Octave embedded in Python. The Synergia framework uses Octave as part of its data analysis suite. Octapy allows Python code to call arbitrary Octave code and transparently exchange values between the two languages. Octapy was developed by one of this paper's authors (Amundson).

PyPar

PyPar [6] is a Python module that provides bindings for a subset of the Message Passing Interface (MPI). One distinct advantage that PyPar has over other Python-MPI solutions is that it does not require a recompilation of the Python interpreter.

CONCLUSION

We have described the software development effort that has been undertaken to in the context of the Synergia beam simulation framework. Currently, most of our effort is focused upon prototype-building and refactoring of the IMPACT library. The end result of this work will be a more flexible and extensible software architecture.

REFERENCES

[1] J. Qiang, R. D. Ryne, S. Habib and V.Decyk, J. Comput. Phys. 163, 434 (2000).

[2] L. Michelotti, FERMILAB-CONF-91-159 presented at 14th IEEE Particle Accelerator Conf., San Francisco, CA, May 6-9, 1991.

[3] http://hifweb.lbl.gov/Forthon.

[4] T. Mens, "Refactoring: Current Research and Future Trends," Preliminary version for LDTA'03 workshop at The European Joint Conferences on Theory and Practice of Software (ETAPS'03), Warsaw, Poland, April 5-13, 2003.

[5] J. Qiang, et al, Parallel Strong-Strong/Strong-Weak Simulations of Beam-Beam Interactions in Hadron Accelerators, Beam Halo Dynamics, Diagnostics and Collimations, J. Wei, W. Fischer, P. Manning, eds., AIP (2003).

[6] http://datamining.anu.edu.au/~ole/pypar.

 0-7803-8859-3/05/$20.00 ©2005 IEEE