SOFTWARE COMPONENTS FOR ELECTRON CLOUD SIMULATIONS

Douglas Ricker Dechow, Peter Stoltz, Tech-X Corp., Boulder, CO 80303, USA
Boyana Norris ANL, Argonne, IL 60439, USA
James Frederick Amundson, Fermilab, Batavia, IL 60510, USA

Abstract

The Synergia2 beam dynamics code is an attempt to incorporate state-of-the-art space charge models from the Impact code into the Chef accelerator tracking code. The need to add new accelerator physics capabilities to the Synergia2 framework has led to software development efforts based on the Common Component Architecture (CCA). The CCA is a specification and a toolset for developing HPC from interchangeable parts, called components.

Electron cloud is a potentially limiting effect in the performance of both high-intensity electron and proton machines. The modeling of electron cloud effects is important for the Fermilab main injector. Here, electron cloud effects are expected to play a significant role when the main injector operates in the regime of a high-intensity proton source for the neutrino program. In the ideal case, computational accelerator physicists would like to be able model electron cloud generation and dynamics in a single, self-consistent simulation. As a first step towards creating component-based, electron cloud generation simulations, this work describes a CCA component created from TxPhysics, a library of impact and field ionization routines.

INTRODUCTION

Electron cloud is one of the most important factors effecting the dynamics of particle beams. Electron cloud is a potentially limiting effect in the performance of both high-intensity electron and proton machines. In addition, the simulation of electron cloud effects pose, at least, two significant computational challenges. The first challenge is due to the multi-species nature of electron cloud simulations. Electron cloud simulations consist of primary beams and electrons. The second challenge results from the large number of particles which are necessary for these simulations.

Computational modeling and simulation of particle accelerators is a foundational tool for understanding the full life-cycle of accelerators: analysis, design, optimization, and upgrading. The Community Petascale Project for Accelerator Science and Simulation (COMPASS) `https://compass.fnal.gov/`, a DOE Scientific Discovery through Advanced Computing (SciDAC-2) program with funding from the Offices of HEP, NP, BES and the Office of Advanced Scientific Computing Research (ASCR), is tasked with developing a "comprehensive set of interoperable components for beam dynamics, electromagnetics, electron cooling, and advanced accelerator modeling." [6]

As a part of the COMPASS project, we are pursuing the development a component-based modeling tool for simulating one of the most pervasive issues in accelerator physics, the electron cloud effect (ECE) [5]. The ECE is an interaction between the particles in accelerator and unwanted electrons that come from residual background gas or the accelerator walls. In some cases, the ECE can cause the number of unwanted electrons to grow exponentially and degrade the quality of the beam in the accelerator to the point that the beam is destroyed. The ECE modeling tool we plan to develop will include components derived from the COMPASS SciDAC codes, Synergia and TxPhysics.

BACKGROUND

The modeling of electron cloud effects is important for projects such as the ILC damping ring and the Fermilab main injector. In the case of the Fermilab main injector, electron cloud effects are expected to play a significant role when the main injector operates in the regime of a high-intensity proton source for the neutrino program. In the ideal case, computational accelerator physicists would like to be able model electron cloud generation and dynamics in a single, self-consistent simulation.

In this section, we discuss Synergia2, the Common Component Architecture (CCA), and TxPhysics.

SYNERGIA2

The Synergia2 application is a hybrid, multi-language, computational science framework developed at Fermi National Accelerator Laboratory (Fermilab) for modeling beam dynamics of high-energy accelerators. The Synergia2 framework tracks the position and velocity of simulated particles as they move along the length of the accelerator. The analytic approximation model for the magnetic force calculations makes use of the *Chef* beam dynamics application developed at Fermilab. Synergia2 also models in a self-consistent way particle space charge forces in three dimensions. Currently, the space charge calculations can be completed by two separate solvers; the Fortran 90 based *Impact* code [7], developed at Los Alamos National Laboratory and presently maintained at Lawrence Berkeley National Laboratory; and the C++-based *Sphyraena* solver which has recently been developed at Fermilab.

Table 1: The implementation languages of the Synergia2 scientific packages and libraries.

Packages	Python	C++	C	F90	F77
Synergia2-Simulations	X				
Synergia2-Sphyraena		X			
Synergia2-IMPACT				X	
Synergia2-Chef		X			
MaryLie/ IMPACT				X	X
TxPhysics			X		

Common Component Architecture (CCA)

The Common Component Architecture (CCA) [1] is a specification and a toolset for developing scientific software from interchangeable parts, components.

The most basic unit of interaction between CCA components is the *port*. A port is analogous to a method in a OO language or a function, procedure, or subroutine in a procedural langauge. CCA port relationships are defined by the Provides/Uses design pattern. As such, they come in two flavors:

1. A **providesPort** defines a protocol that will be supported by the component implementation.
2. A **usesPort** specifies that a component will interact with another component by means of the protocal described in its *providesPort*

Additionally, because Synergia2 is a hybrid software application that is comprised of software libraries and tools which are written in several different languages, multi-language interoperability is of paramount importance to the current and planned work. Table 1 contains a matrix delineating the Synergia2 project packages and libraries and their respective implementation languages that will be used to develop electron cloud simulations.

OASCR has funded the Babel project [4, 2] as a tool for the general language interoperability problem. Unlike other language interoperability solutions, Babel specializes in high performance and in the support of FORTRAN 77 and Fortran 90 through the use of CHASM [8]. In the Babel system, users define their types using the Scientific Interface Definition Language (SIDL). The Babel compiler then generates glue-code for each language. The Babel runtime library provides basic facilities and infrastructure to keep the model consistent. At present, Babel supports FORTRAN 77, Fortran 90, Python, Java, C, and C++ and so removes language interoperability concerns from the developers. An example of SIDL usage is provided in Section .

TxPhysics

Currently, developing ECE simulations usually requires two separate codes in order to capture two distinct physics effects: (i) cloud generation, and (ii) cloud beam dynamics beam. Most simulation codes can model the generation or the dynamics, but not both (WARP+POSINST does model both effects). In normal usage scenarios, the POSINST code models cloud generation, but not beam dynamics. Similarly, the Synergia code models beam dynamics but not cloud generation.

The TxPhysics library provides physics routines used by WARP to handle electrons created by ions striking walls in the simulation. While TxPhysics is written in C, WARP calls the relevant TxPhysics routines through a Python Interface.

RELATED WORK

In a previous project, we prototyped a beam dynamics application based on CCA-compliant software components [3]. As an example of the type of components that were created for the previous project, we will describe the linear beam optics component that was created to wrap services provided by MaryLIE/IMPACT (ML/I). Our work centered on the most basic of optics algorithms, those representing dipole and quadrupole magnets.

The *BeamOpticsPort* port encapsulates tasks related to the application of transfer maps to the particles in the beam. The SIDL code used to encode the *BeamOpticsPort* port abstraction is shown below.

```
interface BeamOpticsPort extends gov.cca.Port
{
void initialize();
void fquad3(in double l,
      in double gb0,
      inout array<double,1,column-major> h,
      inout array<double,2,column-major> mh);

void dquad3(in double l,
      in double gb0,
      inout array<double,1,column-major> h,
      inout array<double,2,column-major> mh);

void drift3(in double l,
      inout array<double,1,column-major> h,
      inout array<double,2,column-major> mh);
}
```

It is essentially a method-to-subroutine wrapping of the ML/I subroutines that provide the services used by the component. The *BeamOptics* component, written in Fortran90, implements the *BeamOpticsPort* port.

CCA/SYNERGIA PROTOTYPE APPLICATION

As a demonstration case for creating a CCA-compliant component from a TxPhysics algorithm, the *get_effective_charge* algorithm was chosen. This effort was successful and is considered to be a first step in a series that will lead to a component-based ECE solution.

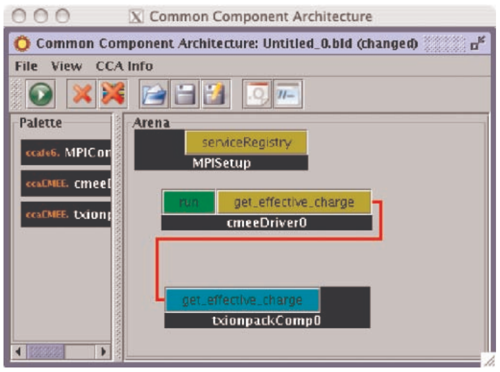

Figure 1: The prototype TxPhysics component connected to its driver.

The prototype component is shown along with its driver component in Figure 1.

ONGOING AND FUTURE WORK

At this point in time, we believe that the realization of a single tool to successfully model the ECE will require the development of no fewer than four coarse-grained, macro components:

- *Synergia.BeamOptics*: an ML/I component for obtaining transfer maps
- *SynergiaCCA.CloudGenComp*: a proposed Synergia2 component for managing the electron cloud particles produced during the generation phase
- *SynergiaCCA.CloudDynComp*: a proposed Synergia2 component for calculated the electron cloud dynamics
- *SynergiaCCA.MacroBeamBunchComp*: a recently developed Synergia2 component for managing the particles of a beam bunch in a 6-D representation
- *SynergiaCCA.SphyraenaSolver*: a recently developed Synergia2 component for solving Poisson's equation

Several of these components, including those such as *SynergiaCCA.MacroBeamBunchComp* that were developed in a prior project, are shown in Figure 2.

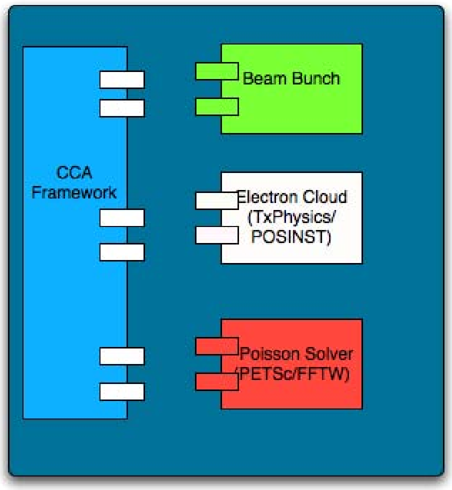
CCA Framework
Beam Bunch
Electron Cloud (TxPhysics/ POSINST)
Poisson Solver (PETSc/FFTW)

Figure 2: CCA/Synergia components for the ECE problem.

REFERENCES

[1] D. E. Bernholdt, B. A. Allan, R. Armstrong, F. Bertrand, K. Chiu, T. L. Dahlgren, K. Damevski, W. R. Elwasif, T. G. W. Epperly, M. Govindaraju, D. S. Katz, J. A. Kohl, M. Krishnan, G. Kumfert, J. W. Larson, S. Lefantzi, M. J. Lewis, A. D. Malony, L. C. McInnes, J. Nieplocha, B. Norris, S. G. Parker, J. Ray, S. Shende, T. L. Windus, and S. Zhou. A component architecture for high-performance scientific computing. *Intl. J. High Perf. Comp. Appl.*, 20(2):163–202, 2006.

[2] T. Dahlgren, T. Epperly, and G. Kumfert. *Babel User's Guide*. CASC, Lawrence Livermore National Laboratory, version 0.10.8 edition, July 2005.

[3] D. R. Dechow, B. Norris, and J. Amundson. The common component architecture for particle accelerator simulations. In *CompFrame '07: Proceedings of the 2007 symposium on Component and framework technology in high-performance and scientific computing*, pages 111–120, New York, NY, USA, 2007. ACM.

[4] Lawrence Livermore National Laboratory. Babel. `http://www.llnl.gov/CASC/components/babel.html`, 2007.

[5] K. Ohmi. *Physical Review Letters*, 75:1526, 1995.

[6] Panagiotis Spentzouris (PI). Community Petascale Project for Accelerator Science and Simulation (COMPASS). FNAL DOCDB, CD-doc-2098, version 1, 2007.

[7] J. Qiang, R. D. Ryne, S. Habib, and V. Decyk. An Object-Oriented Parallel Particle-in-Cell Code for Beam Dynamics Simulation in Linear Accelerators. *Journal of Computational Physics*, 163:434–451, Sept. 2000.

[8] C. E. Rasmussen, M. J. Sottile, S. Shende, and A. D. Malony. Bridging the language gap in scientific computing: The Chasm approach. *Concurrency and Computation: Practice and Experience*, 2005. to appear.