# STAR detector simulations in OO era

Pavel Nevski

November 16, 1998

abstract

The Solenoidal Tracker At RHIC (STAR) is a large acceptance collider detector scheduled to start its operation at Brookhaven National Laboratory in the fall of 1999. The complete STAR detector will contain a set of Time Projection Chambers (TPCs) for charged particle tracking over almost six units about central rapidity, a silicon detector for vertexing, electromagnetic calorimeters, and a number of other systems.

STAR is designed to measure the momentum and identify several thousand particles per event from the collision of heavy nuclei at ultrarelativistic energy. Over 300 Terabytes of real and simulated data will be generated each year. The unprecedented complexity of the events and the total data volume present a formidable software challenge.

Detailed GEANT simulations are a crucial element in detector design and optimization, as well as physics analysis throughout the lifetime of STAR. It is essential to maintain a coherent and complete detector description that is consistent among all elements of simulation and reconstruction software over the lifetime of the experiment. The design of such a system presents a unique challenge, given the ever-changing nature of computing technology.

We are using a hierarchical software design to fullfil this goal. The STAR detector is described in a geometry specification language which can be converted by a parser into Fortran or C++, depending upon the need. The specification language includes a data query mechanism, providing a database interface for detector versioning.

The interface to the current GEANT3 implementation is accomplished by a special Advanced Geant Interface library, serving as an intermediate

software layer. It maintains the GEANT-specific tables of materials, volumes, hit descriptors, etc., and insures the internal consistency of most of the actual parameters of the GEANT routines, thereby significantly reducing the amount of information the user needs to worry about and improving the robustness of the program. The same geometry description can be converted into a set of C++ classes to provide an OO detector model, which can be used with GEANT4 simulation tools and event reconstruction.

In this paper we present our experience with long term system maintainance and providing tools supporting migration to the OO world.