Journal of Physics: Conference Series **513** (2014) 022036 doi:10.1088/1742-6596/513/2/022036

The abstract geometry modeling language (AgML): experience and road map toward eRHIC

Jason Webb, Jerome Lauret and Victor Perevoztchikov
Brookhaven National Laboratory
E-mail: jwebb@bnl.gov

Abstract. The STAR experiment has adopted an Abstract Geometry Modeling Language (AgML) as the primary description of our geometry model. AgML establishes a level of abstraction, decoupling the definition of the detector from the software libraries used to create the concrete geometry model. Thus, AgML allows us to support both our legacy GEANT 3 simulation application and our ROOT/TGeo based reconstruction software from a single source, which is demonstrably self- consistent. While AgML was developed primarily as a tool to migrate away from our legacy FORTRAN-era geometry codes, it also provides a rich syntax geared towards the rapid development of detector models. AgML has been successfully employed by users to quickly develop and integrate the descriptions of several new detectors in the RHIC/STAR experiment including the Forward GEM Tracker (FGT) and Heavy Flavor Tracker (HFT) upgrades installed in STAR for the 2012 and 2013 runs. AgML has furthermore been heavily utilized to study future upgrades to the STAR detector as it prepares for the eRHIC era.

With its track record of practical use in a live experiment in mind, we present the status, lessons learned and future of the AgML language as well as our experience in bringing the code into our production and development environments. We will discuss the path toward eRHIC and pushing the current model to accommodate for detector miss-alignment and high precision physics.

1. Introduction

The STAR experiment has migrated its geometry model away from its legacy framework, adopting an Abstract Geometry Modeling Language (AgML)[1] which allows us to support both our GEANT 3[2] based simulation codes and our reconstruction chains which utilize the ROOT/TGeo[3] geometry framework. The primary motivation of AgML was to enable a consistent single-source description of our detector geometry shared by our simulation and reconstruction packages, while enabling us to better leverage modern virtual Monte Carlo toolkits. AgML furthermore retains and extends functionality from the original AgSTAR framework[4], which promotes ease-of-use, rapid development and code maintainability. Following extensive regression tests to validate the new geometry model[1], STAR integrated AgML into its simulation and reconstruction chains in 2012. The new geometry model currently supports our simulation and event reconstruction needs, providing consistent and stable reference geometries for the majority of STAR's run configurations. We have integrated two major new subsystems into our software stack using the AgML framework: the Forward GEM Tracker (FGT), and the silicon pixel detector (PXL) of the new Heavy Flavor Tracker (HFT). AgML is additionally being used to evaluate upgrade paths for eSTAR, a planned experiment at an

electron-ion collider at RHIC. In this report we will highlight those features of the language we feel are useful for both production and development environments, discuss the capabilities and plans for the AgML language framework, and summarize our experience in integrating new subsystems into the STAR geometry model.

2. Language

AgML is an XML-based language geared towards the problem domain of generating geometry models for nuclear and high-energy experiments. The language implements syntax for declaring the properties, content and placement of volumes within the geometry tree, as well as markup elements for data structures, iteration and flow control. This enables the complete specification of detector geometries within single XML files, called modules. Listing 1 shows the overall structure of a detector module. Data structures are declared and filled, materials defined and the top-level volume of the module is created and placed in the geometry tree. Volumes are declared as complete code blocks in XML. The properties of the volume are specified within this code block, as well as the creation and placement of daughter volumes. This enforces a uniform organization of modules which we have found to be particularly useful in developing and maintaining the code. The complete behavior of a volume can be determined by referencing a fairly compact block.

```
<Module name="BBCM" comment="The Beam Beam Counter Module"  >
...
Data-Structures
Material-Definitions [optional]
Create-and-Place-Top-Volume
...
<Volume name="THXM" comment="is one Triple HeXagonal Module"  >
  <Material name="Air"  />
  <Medium   name="standard"  />
  <Attribute for="THXM" seen="0" />

  <Shape type="tube" dz="HEXG.thick/2"
                     rmax="HEXG.irad*2.0/sin(pi/3.0)" />

  <For var="j" from="0" to="2"  >
    rsing=HEXG.irad/sin(pi/3.0)
    thesing= j * pi * 2.0/3.0
    <Create     block="SHXT"  />
    <Placement  block="SHXT" in="THXM"
                x="rsing*cos(thesing)"
                y="rsing*sin(thesing)" />
  </For>

</Volume>
...
</Module>
```

Listing 1. AgML syntax describing the triple hexagonal module of the STAR Beam-Beam counters.

The developer's geometry modules are responsible for steering the creation of the detector geometry through making calls to functions defined in a support library. The support library interfaces with the concrete geometry packages, such as ROOT/TGeo[3] or AgSTAR/GEANT 3[4, 2], which then instantiates the concrete geometry model. While the language syntax enforces a useful organization of the user's modules, the support library adds additional functionality

which streamlines the development cycle, promotes ease-of-use and improves maintainability of the code. Some of the most important features in this regard are:

Inheritance
The rules enforced by the support library requires only that the type of shape is specified in a volume. All other parameters, e.g. the material, medium and even the dimensions of the shape, may be omitted in the definition of the volume block. The resulting behavior is that the volume being created will inherit its properties from its mother volume. This arrangement provides a mechanism by which developers can quickly create branches of the geometry tree with dimensions tied to each other in a way which is automated by the support library. Changes to one volume will be reflected in all branches beneath than volume which inherit parameters from it. This is particularly useful for optimizing detector designs, simplifying the process of trying out different detector configurations and minimizing overlap and extrusion errors when making modifications to the geometry.

Volume Parameterization
Volume parameterization is another useful feature which is supported by the AgML libraries. An individual volume block in an AgML module may be used to create multiple versions of a concrete volume. When a volume is referenced by the create operator, the support library examines the previous states of that volume block. If it finds a volume with identical parameters (i.e. shape, material and medium), then the existing volume will be used along with all of the daughter volumes. If one or more of the volume's parameters have been changed, a new volume (with a unique name) will be created, and the geometry tree will be traversed again to create and place new daughter volumes. This feature allows a single compact code block to generate several similar detector elements.

Data Versioning
AgML data structures enable a single module to easily support multiple configurations of a detector. Different versions of the data structure may be filled, selected at run time, and used to steer the creation of the geometry model. STAR has typically staged the installation of major subsystems over several runs. Data versioning allows us to support the changing state of a detector using a single file. This feature greatly simplifies supporting earlier versions of the STAR detector: as the experiment progresses, and the geometry model is refined, those improvements are immediately available. The task of optimizing detector designs is also streamlined by this feature, as many concepts can be defined in a single file, selected and run time, and tested.

Steering
Since each geometry module supports multiple versions of a subdetector in STAR, a mechanism is needed to steer the creation of a specific run's geometry. AgML provides facilities to manipulate a module's data structures from a main steering routine. Thus, we pass configuration information from the main geometry module to the modules which create each subdetector through data structures.

3. Existing Framework and Planned Extensions

AgML provides an adaptable framework for creating concrete geometry models. Figure 1 illustrates the generation of the concrete geometry. AgML source files are parsed and converted into code (e.g. C++, FORTRAN) through one of two extensible backends to the AgML parser[1]. This code is compiled and linked in with a dedicated support library, through which it steers the creation of the concrete geometry model. We currently support two such models: GEANT 3 and ROOT/TGeo. New functionality can be added to the AgML language by adding to the backends, and implementing the corresponding code with the support libraries. For example, in the next release of AgML we plan to provide support for the full set of features available in ROOT

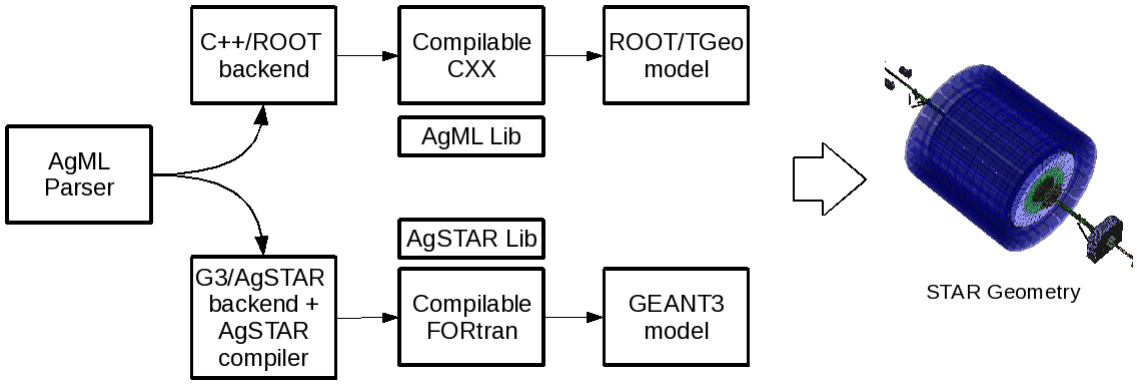

Figure 1. Geometry workflow. AgML files are parsed and passed through one of two backends, producing compilable code. These codes are linked with the support library for the concrete geometry and an additional steering routine, capable of instantiating one the various configurations of the STAR detector using either GEANT 3 or ROOT.

geometry model. At present, only those features in common with GEANT 3 are supported. We also intend to add support for Geant 4[5] geometries[1], and are considering providing a GDML[6] backend as well.

With the installation of the FGT and the HFT, STAR is entering a new era of high-precision physic measurements. This places new demands on our geometry model, as well as our simulation and event reconstruction environments. On the reconstruction side, we are prepared to support detailed misalignments of the detector. AgML automatically inserts "alignment groups" (volume assemblies) in the concrete ROOT geometry, which groups together co-moving geometry volumes. These alignment groups can then be misaligned by the reconstruction chain, based on parameters stored in our offline database. For simulations, additional work needs to be done as GEANT 3 does not provide the same support for misalignments as is found in ROOT. The AgSTAR/GEANT 3 support library will be extended with this missing functionality, adding support for alignment groups and application of the corresponding alignment constants from, for example, a database.

4. Experience

STAR is pursuing an upgrades program which will extend the physics reach of the experiment, and is planning for its evolution to the eSTAR detector at an electron-ion collider at RHIC. New detector models are being implemented by user groups in STAR, with assistance from the simulations group, in order to support the new detectors and upgrade proposals. Typically the work of developing geometries is assigned to a graduate student or postdoctoral researcher working within the group which is responsible for the construction of the detector, under the supervision of more senior personnel. Figure 2 illustrates two detectors which have been integrated into the experiment and the geometry framework: the Forward GEM Tracker (FGT), fully installed for the y2013 run; and the silicon pixel detector of the Heavy Flavor Tracker, partially installed in y2013. These detectors are both designed to provide high resolution tracking in high-multiplicity environments. Thus, their geometry models are quite detailed. In each case, the time required for postdoctoral physicists with no prior experience to develop and validate these production quality AgML geometry models was about 1-2 FTE months.

[1] This would require some reshape of the STAR geometry model to eliminate overlapping volumes.

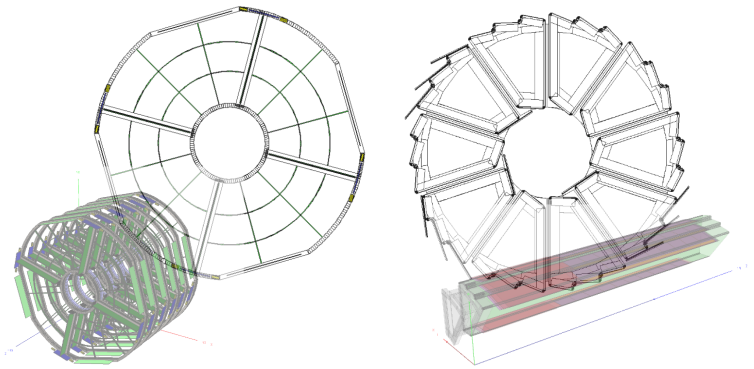

Figure 2. Forward GEM Tracker (left) and the silicon pixel detector of the Heavy Flavor Tracker (right). The line drawing is of a single FGT disk, and the arrangement of the six disks of the FGT is illustrated. The pixel detector shows a line drawing of the organization of the pixel ladders on the ten sectors for the y2013 run, while the overlay shows a single instrumented sector.

5. Summary

The geometry model is an essential element of data processing applications in nuclear, particle and high-energy physics experiments. STAR has successfully integrated the Abstract Geometry Modeling Language into our production and development environments, providing a robust single-source description of our detector, capable of supporting both our legacy GEANT 3 simulation application, and our ROOT-based reconstruction codes. The AgML framework provides a feature-rich, compact description of the STAR geometry model. It is an extensible framework, which will allow us to add missing functionality to our legacy GEANT 3 simulations, and provide support for new concrete geometry models[7] if and when they become available in the field. AgML is being employed successfully in our production environment, integrating new detectors (FGT and HFT) for the 2012 and 2013 runs at RHIC. And it is serving well in our development environment, supporting the simulations of the proposed upgrade path to eSTAR.

References

[1] J. Webb et al., *Planning for Evolution in a Production Environment: Migration from a Legacy Geometry Code to an Abstract Geometry Modeling Language in STAR*, J. Phys. Conf. Ser. 396 022058 (2012).
[2] *GEANT - Detector Description and Simulation Tool*, CERN, Geneva 1993.
[3] R. Brun et al., *The ROOT Geometry Package*, Nucl. Instrum. Meth. **A502**:676-680,2003.
[4] A. Artamonov et al., *DICE-95*, internal note ATLAS-SOFT/95-14, CERN, 1995; P. Jacobs and D. Irmscher, *GSTAR: A Geant-based Detector Simulation Chain for STAR*, STAR internal note 0235, see also https://drupal.star.bnl.gov/STAR/starnotes/public/sn0235.
[5] S. Agostinelli et al., *Geant4 - a simulation toolkit*, Nucl. Instrum. Meth. **A506** (2003) 250-303.
[6] R. Chytracek, *The Geometry Description Markup Language*, Conf. Proc. C **0109031** (2001).
[7] *GEANT project*, http://geant.cern.ch