STAR VMC project

Maxim Potekhin for the STAR Collaboration

VMC workshop at CERN

Nov. 29-30 2004

Overview of the STAR VMC effort

- Motivations for the transition from the current legacy system to VMC have been presented at CHEP'03 and 04.
  - STAR as a running experiment has an estimated lifetime of approximately 10 years, hence an urgent need to address the maintainability issue
  - Reliable production is a priority, for both experimental data and Monte Carlo. There is an extensive body of reconstruction software that must be interfaced with by any new simulation system, and the issue of shared components must be resolved
  - There are, and will be, ongoing upgrades that require simulation and geometrical modeling. New tools and approaches are needed to make the upgrades successful
- Evolution of the STAR simulation software is a necessity rather than a choice

Overview of the STAR VMC effort

- We maintain focus on the creation of a single source, unified geometry model for simulation and reconstruction
- We are borrowing from the Alice VMC experience
- In addition, attention is being paid to
  - Interface with the existing reconstruction software in STAR (how do we pass the data?)
  - Tools for the Detector Description
  - Hit structure declaration (sensitive detector description)
  - Visualization (work by V.Fine)
  - User-prescribed volume enumeration

Overview of the STAR VMC effort

- What do we bring to the VMC table?
  - as a running experiment, we are in the position to fully test VMC in all its aspects, including the geometry model and integration with reconstruction software
  - will test ROOT geometry as applied in reconstruction
  - we are interested in the development of the Detector Description and visualization tools, and have resources to do work in that direction
  - we would like to continue collaboration and exchange of ideas with the VMC developers and users, and believe that this would be beneficial for everyone
- In the following, we will consider just two aspects of the VMC:
  - code base
  - geometry model and Detector Description

VMC Code Base

- We are using the TGeant3 class, which currently includes the GEANT 3.21 library
- Obviously, TGeant3 depends on ROOT
- There are other critical applications (such as the main reco engine named root4star) that have dependencies on ROOT and GEANT, and which do not necessarily run the DEV version of ROOT
- There are other critical applications (such as the main simu engine named starsim) that depend on GEANT but not ROOT

  - root4star sometimes needs to load GEANT but not always TGeant3
  - starsim needs to load GEANT and never needs ROOT
  - we would like to use the same version of GEANT in both
  - can't have a ROOT dependency in strarsim

- Solution: factoring out the GEANT 3.21 library? Why not?
- Makes a lot of sense from the software engineering point of view as it improves modularity and allows STAR to run its software in a sensible way

VMC Code Base

- Decoupling TGeant3 and GEANT 3.21:
  - The main idea is to use a simple bridge class to fully decouple those. This new class is called G3BridgeApp, and it contains wrappers for TGeant3 functions that were previously called from geant3. This is accomplished by having function pointers to those, and storing them in the container class G3Bridge, which is used in G3BridgeApp.
  - If a pointer is loaded, the corresponding callback is done, as in this example: void gukine_() {if (gBrd.m_gukine) (*(Sub0)gBrd.m_gukine)();}
  - A few minor functions moved to GEANT 3.21 as there is no dependency on them in TGeant3

Geometry

We plan to employ the ROOT geometry in the STAR detector model for the following reasons:

- ROOT already being a critical components of the STAR software
- ROOT geometry is feature-rich and suitable for complex systems
- proven in GEANT-like applications such as VMC
- can be used in event displays and potentially offers sophisticated graphic functionality
- can be considered “platform neutral” as it is not really tied to any particular application such as GEANT, and can be productively used to navigate geometry in STAR reconstruction
- can provide a consistent geometry interface with reconstruction
- more flexible, and portable, geometry description: geometry persistent in ROOT files, XML, etc

Geometry

- We will pursue a structured solution for the STAR Detector Description, and try to avoid, if possible, using C++ code as primary source for the following reasons:
  - Based on our experience, we believe that having a formal geometry description in a declarative language (similar in idea to what we have now), will facilitate the transition to the new platform
  - Formally structured document is superior in terms of maintenance (can use formal editing and structure visualization tools) – see examples below
  - Having the geometry source outside ROOT opens other possibilities such as interfacing other detector visualization systems not necessarily tied to ROOT

XML-based solution?

- Using XML-based solution can facilitate the geometry development cycle, as a broad array of tools is available to edit, validate and visualize the structure of XML documents. The developers and users are not tied to a particular framework such as G4 or TGeo, and via XML transforms, can benefit from geometry visualization and other tools.

- Experience with XML so far:
  - we don't care what particular schema is used as long as it's adequate
  - coming from the ROOT angle, what does GDML have to offer?
  - observe that XML was used in CMS and LHCb experiments, etc
  - observe that XML detector description was discontinued at ATLAS

XML-based solution?

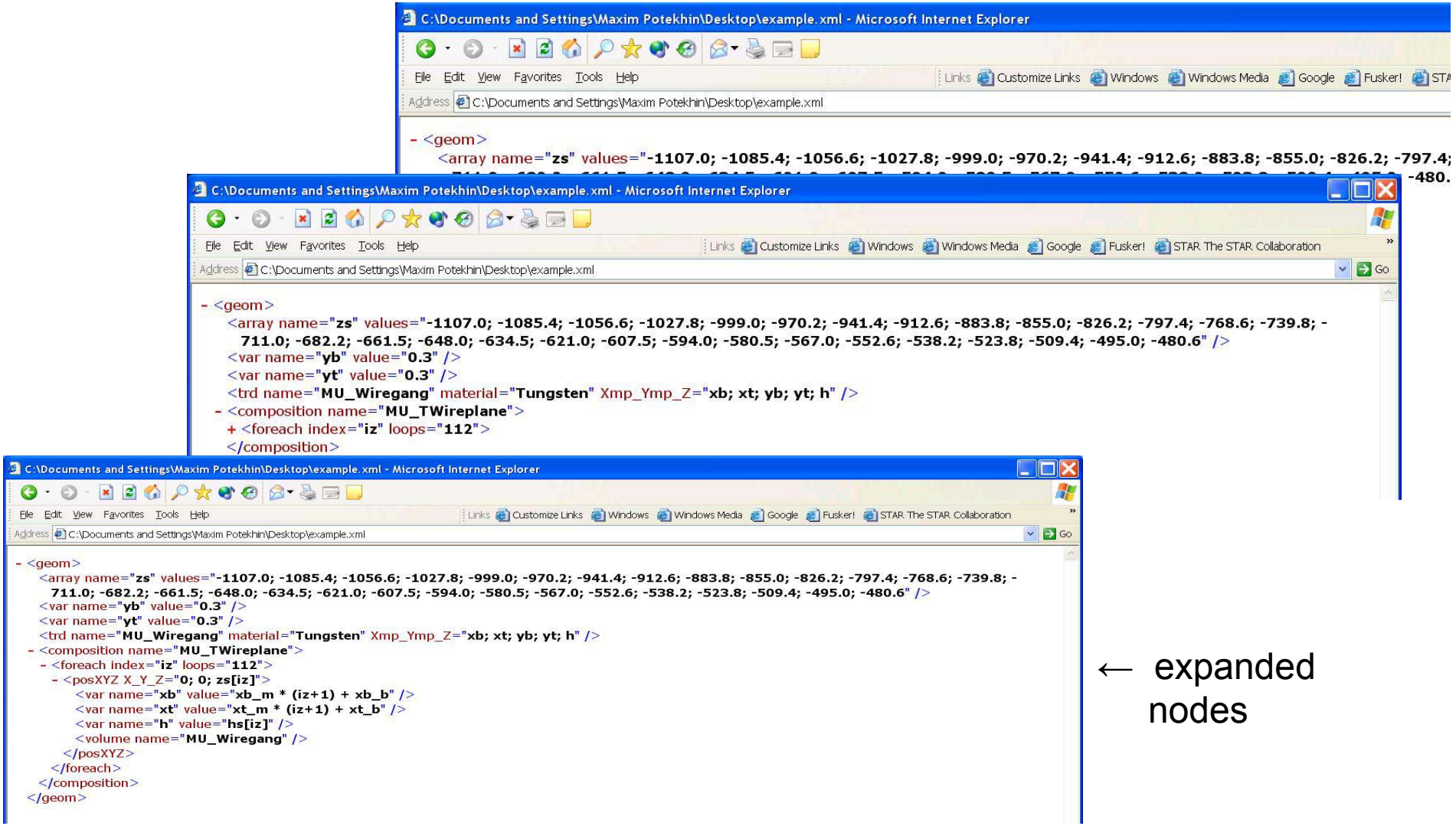
```
<array name="zs" values="    -1107.0;  -1085.4;   -1056.6;  -1027.8;  -999.0;
                              -970.2;   -941.4;    -912.6;   -883.8;  -855.0;
                              -826.2;   -797.4;    -768.6;   -739.8;  -711.0;
                              -682.2;   -661.5;    -648.0;   -634.5;  -621.0;
                              -607.5;   -594.0;    -580.5;   -567.0;  -552.6;
                              -538.2;   -523.8;    -509.4;   -495.0;  -480.6"/>

<var name="yb" value="0.3"/>
<var name="yt" value="0.3"/>
<trd name="MU_Wiregang" material="Tungsten" Xmp_Ymp_Z="xb; xt; yb; yt; h" />

<composition name="MU_TWireplane">
    <foreach index="iz" loops="112">
        <posXYZ X_Y_Z="0; 0; zs[iz]">
            <var name="xb" value="xb_m * (iz+1) + xb_b"/>
            <var name="xt“  value="xt_m  * (iz+1) + xt_b"/>
            <var name="h" value="hs[iz]"/>
            <volume name="MU_TM1_T1_L1_Wiregang"/>
        </posXYZ>
    </foreach>
</composition>
```

XML?

XML is presentable in a variety of clients including the IE. Emphasizing the structure via the layout can make editing and maintenance easier:



← expanded nodes

C++?

```
for (i = 0; i < (nLayers-1); i++ )  {
    label = "XU" ;
    label += i ;
    Float_t tseg ;
    tseg = geom->GetECScintThick()+geom->GetECPbRadThick();
    envelopA[5] = envelopA[6] ;                          // rmin at z1
    envelopA[4] = geom->ZFromEtaR(envelopA[5] + tseg,
                                  geom->GetArm1EtaMin());     // z coordinate 1
    envelopA[7] = geom->ZFromEtaR(envelopA[5] + tseg,
                                  geom->GetArm1EtaMax());     // z coordinate 2
    envelopA[6] = envelopA[5] + tseg ;                   // rmax at z1
    envelopA[8] = envelopA[5] ;                          // radii are the same.
    envelopA[9] = envelopA[6] ;                          // radii are the same.

    gMC->Gsvolu(label.Data(), "PGON", idtmed[1599], envelopA, 10);

    // Position XUi in XEN1
    gMC->Gspos(label.Data(), 1, "XEN1", 0.0, 0.0, 0.0, idrotm, "ONLY") ;
```

Geometry

Successful XML implementation is characterized by the following set of parameters:

- A sufficient assortment of primitives *(solids)*, allowing the user to describe complex geometries. In addition, a straightforward mapping to the solids that exist in the popular simulations engines such as GEANT 3 (further referred to as G3), GEANT 4 (G4) and the Root geometry (TGeo), is preferred. It is sometimes called *solids compatibility*
- Self-contained, single source detector description, i.e. the XML geometry specification should be self contained and not involve significant pieces of structural information coded separately and in a different language.
- Support for the inherent symmetry in the structure being described, i.e. being able to describe identical entities being positioned multiple times, while only describing the underlying volume once and *positioning the copies implicitly*, according to a rule.
- Volume parameterization
- Facilities that support arithmetic expressions
- A possibility to separate the structural component (such as the topology of volume hierarchy) from the numerical data component (such as volume positions and sizes). For the purposes of “primary source”, static XML code is not adequate as the “conditions” data will typically be maintained in an external database

Geometry

Is GDML the right tool for us? It is instructive to compare GDML and some other XML implementation of the Detector Description system, such as AGDD

Let us follow criteria listed on the previous page:

- *Solids compatibility*: by and large, both are adequate for the task
- *Self-contained source*: yes in both (version 6 of AGDD)
- *Volume parameterization*: different approach, similar capability
- *Facilities that support arithmetic expressions*: “yes” in both
- *Numerical data component:* currently “no” in both

Differences - the following are unique in AGDD, in the area of symmetry:

- Arrays
- Iterators
- Multiple positioning operators

GDML is not suited to be a “primary source” geometry description medium as is lacks in the area of symmetry and database interface. It is currently considered an “exchange” format and presents a serialized and static code.

What’s the next step?

Conclusions

- STAR is working on a transition from a legacy simulation software to a modern platform which is likely to be VMC based
- Being an active (and busy) experiment creates a set of requirements and conditions
  - interface with existing reconstruction software
  - ease of writing up and maintaining the Detector Description: users should be available to start work on the new description right away, using visualization tools available (and not necessarily ROOT-based)
  - STAR a motivated collaborator in the VMC effort, willing to contribute resources to development, testing, and performance enhancement
- ROOT geometry a likely candidate for the geometry model in both simulation and reconstruction
- Detector Description is on the critical path (see above), hence our interest in a structured geometry description, possibly XML based – still looking for the right schema