Geometry framework status

Juan P. Palacios
CERN

Working definition

- Provides users of detector geometry description with geometrical information derived from both a nominal alignment of all the sub-components of LHCb, and deviations from the nominal alignment
- Provides a mechanism for users to modify the deviations from the nominal alignment for a given sub-component and ensures that the modifications are propagated coherently to the geometry description
- Provides a mechanism to write a new set of deviations to the conditions database

(Current) working definition / 2

- **Says nothing about alignment strategies, algorithms or their implementation**
- **Provides the basic functionality needed for such algorithms to function**
- **This definition could evolve**
  - Commonality between different alignment algorithms/tasks?
- **“Alignment framework” misleading, “Geometry framework” a bit general**
- **Maybe we should think of (yet another) name for this…**

Usage of detector geometry service

- **Typically, calls to DetectorElement::geometry() methods**
- **Returns IGeomertyInfo* which has local to global and global to local transformations and a host of other services**
  - See **http://lhcb-release-area.web.cern.ch/LHCb-release-area/LHCB/doc/html/class_i_geometry_info.html**
- **Will keep this part of the interface. IGeometryInfo will now have misalignment information**

Detector element approach

- **Basic idea:**
  - Make transformations accessible to the Detector Element and let it do the work
- **Two limits:**
  1. One detector element per alignable object. Needs basically one transformation.

     THIS IS MY FAVOURED APPROACH TAKEN FOR ALIGNMENT
  2. A detector element that knows enough about its PVolume daughters (and their daughters, and their daughters' daughters, and their daughters' daughters' daughters, and …) to assign the correct transformations to each

     MESSY IN ALIGNMENT CONTEXT, BUT MAYBE NECESSARY FOR OTHER CONDITIONS LIKE CALIBRATIONS?

     SHOULD BE IMPLEMENTED IN EACH DETECTOR'S SPECIALISED DETECTOR ELEMENT

The basic idea

- **One detector element per alignable object**
  - Can be simply detelemfer in XML and default detector element
  - For VELO: whole VELO, detector halves, r-ϕ pairs, individual sensors
  - Requires associated logical volume
- **One “delta” transformation matrix for each detector element**
  - Catalogue of alignment conditions with one to one mapping to alignable objects
    - At the moment test XML version
    - Integrating use of conditions data store interface
  - Delta held in new **AlignmentCondition** object
- **No changes to LHCb detector geometry description philosophy**
- **Few changes to DetDesc and DetDescCnv code**

Implementation status: GeometryInfo

- **Let IGeometryInfo handle both ideal and delta transformations:**
  - Keep same DetectorElement public interface.
  - IGeometryInfo toGlobal and toLocal methods now deal with combined ideal + delta transformations
  - Methods to get ideal geometry have been added
  - Possible for users to update delta matrix
  - Easy to “refresh” state of new GeometryInfo. All caching controlled from one method, aptly named cache()
  - GeometryInfo accesses CondDB (or test XML file), generates all necessary matrices
- **Implementation ready and tested with XML conditions catalogue**

What does GeometryInfo do?

- **Constructor uses path to get AlignmentCondition**
- **Scans down tree picking up parent transformations**
  - Iterative procedure finding GeometryInfos of support detector elements
  - Stores ideal and delta transformations for each level in vectors
    - Allow to re-calculate after updates
  - Combines ideal matrices to get ideal case local to global
  - Combines all matrices to get local to global with misalignments
- **"Local" misalignment matrix can be updated by user**
  - Re-calculates "global" matrix automatically
  - At the moment this is de-coupled from automatic update mechanism

Implementation status: conditions catalogue

- **Test conditions in XML file.**
- **Added “condition” attribute to geometryinfo in structure.dtd**
  - Only necessary to add an

    ```
    <geometryinfo condition=“somePath” />
    ```

    in each <detelem …. /> definition
  - Then store the condition somewhere:

```
<condition classID=“6” name=“somePath”>
     <paramVector name=“dPosXYZ” type=“double”> 0. 0. 0.</paramVector>
     <paramVector name=“dRotXYZ” type=“double”> 0. 0. 0.</paramVector>
</condition>
```

Creation of an AlignmentCondition

- **At the moment, via XML and dedicated “converter”**
- **classID = 6 maps to XmlAlignmentConditionCnv**
  - This is a little template which simply instantiates the right kind of condition
- **AlignmentCondition has access to the parameters and constructs the matrices**

```
<condition classID=“6” name=“/dd/Conditions/LHCb/myDetector/Module67”>
     <paramVector name=“dPosXYZ” type=“double”> 0. 0. 0.</paramVector>
     <paramVector name=“dRotXYZ” type=“double”> 0. 0. 0.</paramVector>
</condition>
```

- **GeometryInfo accesses condition via data service:**

```
SmartDataPtr<AlignmentCondition> cond(datasvc(),
                    “/dd/Conditions/LHCb/myDetector/Module67”);
```

How to use misalignment information?

- **Currently the “ideal” transformations are obtained from the geomerty() methods of the DetectorElement PI.**
- **Keep this interface**
  - DetectorElement::geometry() still points to IGeometryInfo
  - IGeometryInfo handles ideal and misaligned geometry
  - Standard transformation methods deal with misaligned geometry
    - User code will automatically get misaligned geometry transformations if unchanged
  - New methods allow to perform ideal transformations
  - Delta matrix also available – and can be modified

Requirements from detectors

- **A detector element per alignable object**
  - Must have path to an alignment condition
  - In current scheme this means a condition in the XML <geometryinfo /> definition of the detector element. This contains the correct path in the data store
- **A catalogue of conditions with one-to-one mapping to detector elements**
  - Actually, in absence of condition identity matrix is created so current XML descriptions still work for ideal
- **A geometry description where the alignable objects can be associated to an LVolume**

Summary: what is there

- **XML conditions catalogue**
  - Changes to structure.dtd condition definition and XmlBaseConditionCnv to allow for conditionrefs inside conditions
  - VELO example tree of references
    - Working for now with dummy XML alignment parameter vectors
- **New GeometryInfo class**
  - New implementation
  - Instantiated in DetectorElement from XmlBaseDetElemCnv
  - Modifications to both the above to pass condition address
- **AlignmentCondition class**
  - Assigned **classID 6** and "Wrote" XmlAlignmentConditionCnv
  - Instantiated in GeometryInfo using path to get alignment parameters from (XML based for now) data store.

Work in progress

- **Use of CondDB data store interface**
  - Conditions retrieval
  - Updating conditions in data store and DB
- **Incorporate update manager**
  - Mechanism to update necessary cached info after a condition has changed.

Have necessary code from Marco. Incorporate this week.

- **Update of conditions on the fly:**
  - Implemented, needs testing
- **Visualisation**
  - Could we see move volumes in Panoramix? Good for debugging/interpreting
- **Related note: what about simulation? Can the misalignments be used there?**