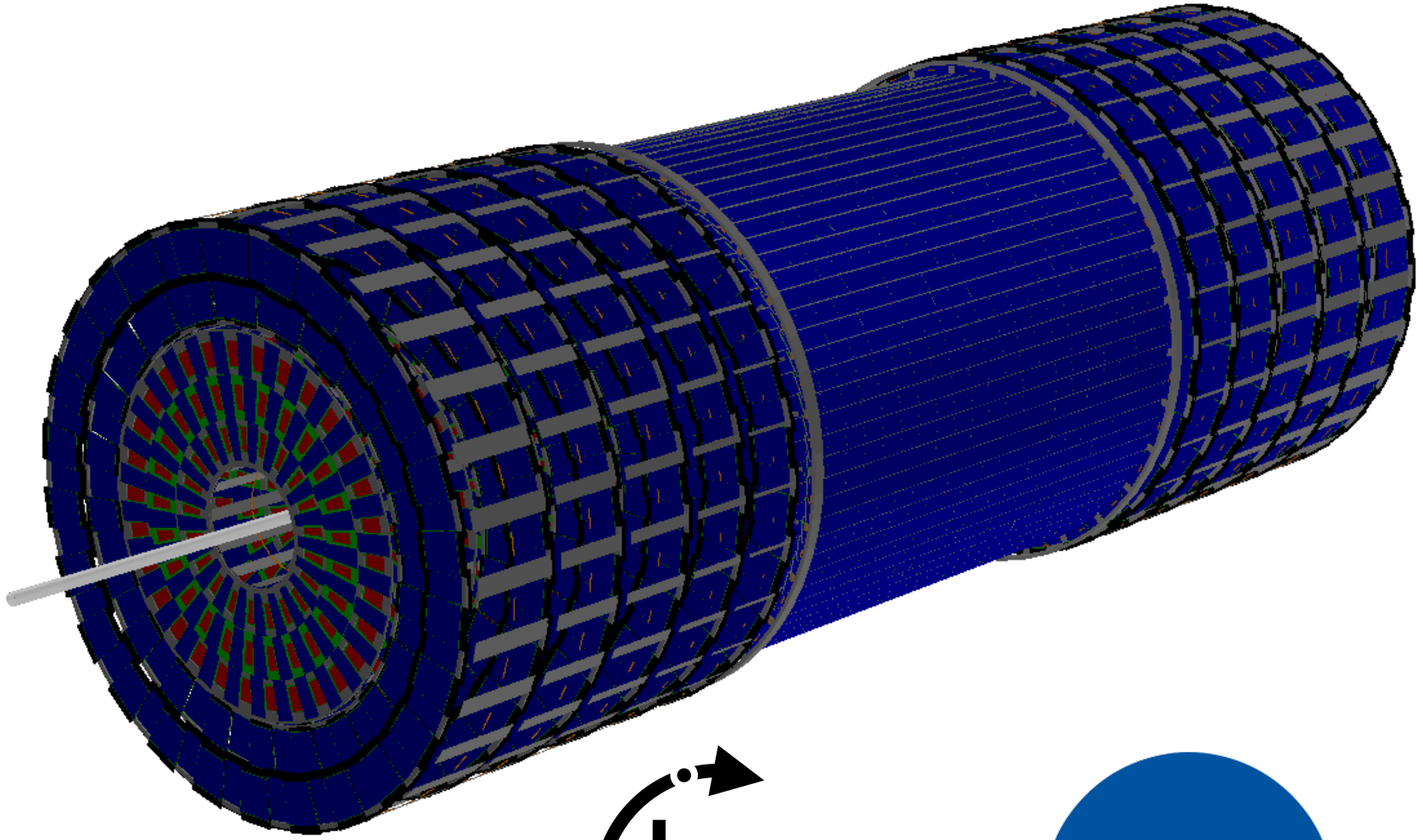
A. Salzburger (CERN)
for the ACTS project

OpenDataDetector (based on TrackML detector) in DD4hep



acts meets DD4hep

supported by

EP-R&D
Programme on Technologies for Future Experiments

iris hep

Bundesministerium für Bildung und Forschung

AIDA innova

cooperations

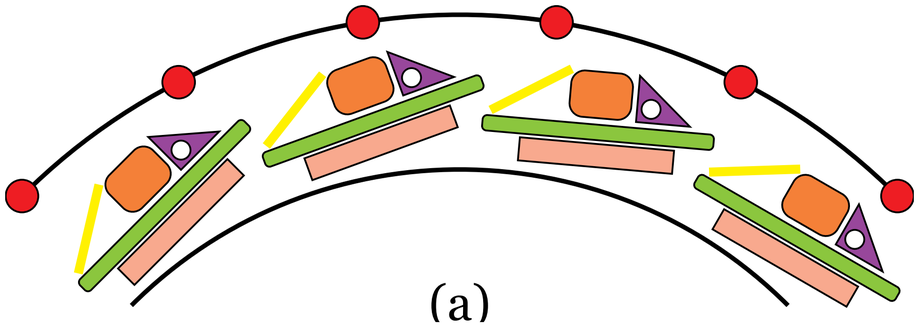
HSF
HEP Software Foundation

CERN openlab

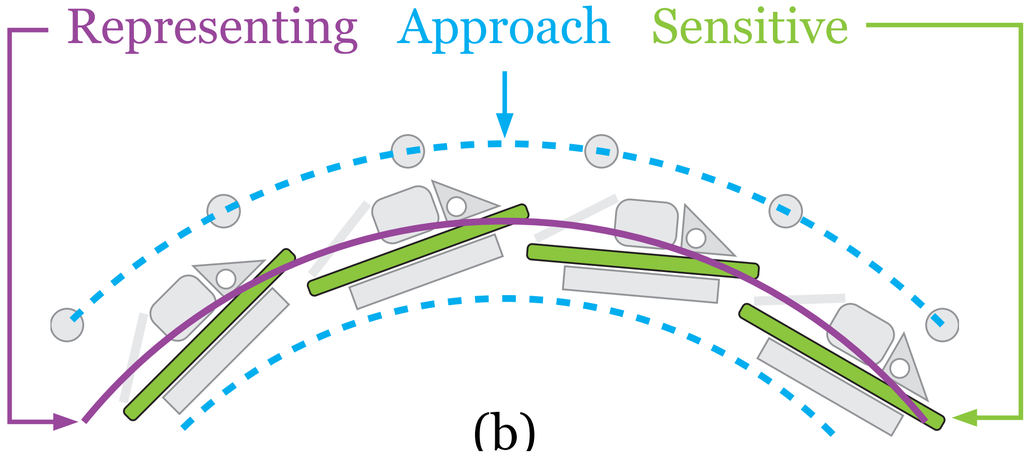
ACTS geometry

ACTS TrackingGeometry is a surface based geometry that implements an intrinsic navigation

- Surfaces between attaching volumes act as portals between them in order to minimise navigational search



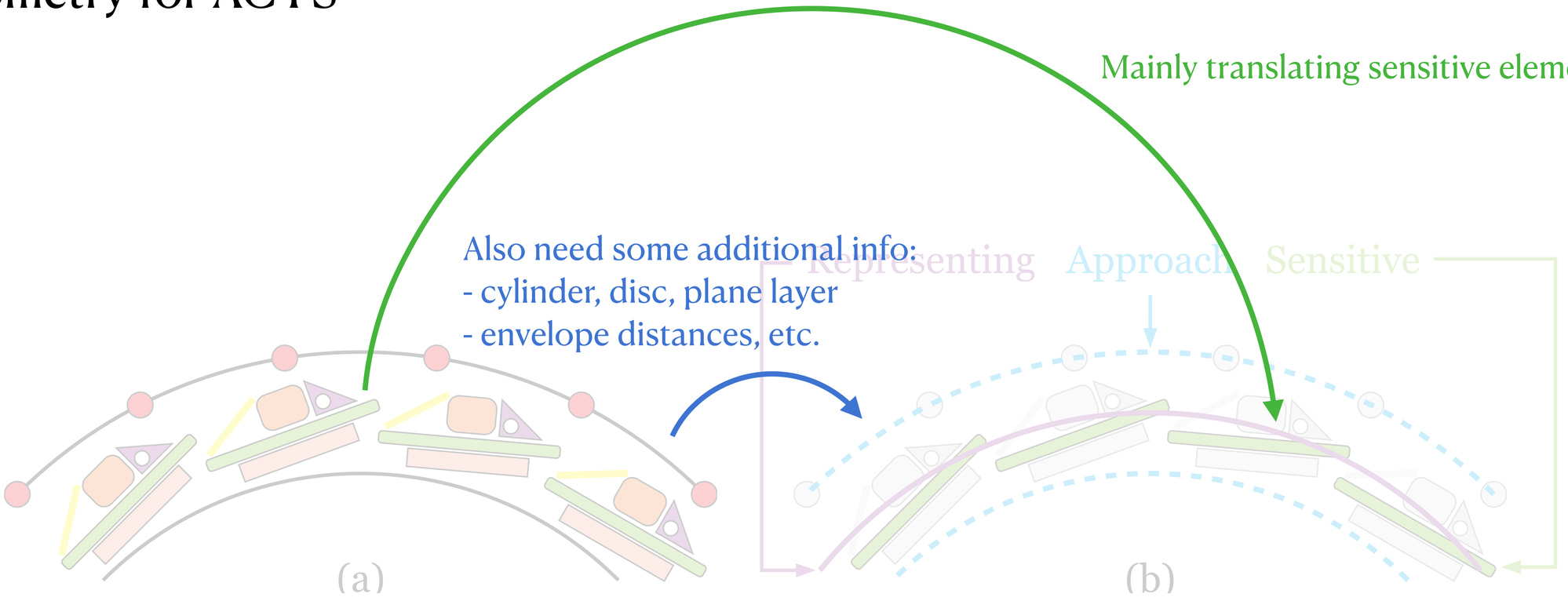
Detailed Geometry
(DD4hep, Geant4, GeoModel, TGeo ...)



Acts::TrackingGeometry

ACTS geometry from DD4hep

A dedicated `Plugin/DD4hep` converter is available to interpret the DD4Hep geometry for ACTS



Detailed Geometry
(DD4hep, Geant4, GeoModel, TGeo ...)

Acts::TrackingGeometry

Acts::ActsExtension object

We attach a dedicated `Acts::ActsExtension` object to DD4hep

- **works, BUT creates an explicit dependency**
- **some examples follow:**

```
static Ref_t TrackerEndcap_o2_v06_geo(Detector& theDetector, xml_h e, SensitiveDetector sens)  {
    typedef vector<PlacedVolume> Placements;
    xml_det_t   x_det     = e;
    Material    vacuum    = theDetector.vacuum();
    int         det_id    = x_det.id();
    string      det_name  = x_det.nameStr();
    bool        reflect   = x_det.reflect(false);
    DetElement  sdet        (det_name,det_id);

    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code start %%%%%%%%%%%%%%%%%%%%%%%%
    Acts::ActsExtension* endcapExtension = new Acts::ActsExtension();
    endcapExtension->addType("endcap", "detector");
    sdet.addExtension<Acts::ActsExtension>(endcapExtension);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code end %%%%%%%%%%%%%%%%%%%%%%%%%%
```

FCCeeCLD

Acts::ActsExtension object

```
for(size_t ic=0; ic<sensVols.size(); ++ic)  {
    PlacedVolume sens_pv = sensVols[ic];
    DetElement comp_elt(module,sens_pv.volume().name(),mod_num);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code start %%%%%%%%%%%%%%%%%%%%%%%%
    Acts::ActsExtension* wafer = new Acts::ActsExtension();
    wafer->addType("passive", "material");
    wafer->addType("axes", "definitions", "XYZ");
    comp_elt.addExtension<Acts::ActsExtension>(wafer);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code end %%%%%%%%%%%%%%%%%%%%%%%%%%
    comp_elt.setPlacement(sens_pv);
}
```

FCCeeCLD

Particularly important:

- how does the volume coordinate frame map into the surface coordinate frame ?

```
for(int k=0; k<nmodules; ++k) {
    string m_base = _toString(l_id,"layer%d") + _toString(mod_num,"_module%d") + _toString(k,"_sensor%d");

    double x = -r*std::cos(phi);
    double y = -r*std::sin(phi);
    DetElement module(sdet,m_base+"_pos",det_id);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code start %%%%%%%%%%%%%%%%%%%%%%%%
    Acts::ActsExtension* sensorExtension = new Acts::ActsExtension();
    sensorExtension->addType("sensor", "detector");
    sensorExtension->addType("axes", "definitions", "XZY");
    module.addExtension<Acts::ActsExtension>(sensorExtension);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code end %%%%%%%%%%%%%%%%%%%%%%%%%%
```

FCCeeCLD

Acts::ActsExtension object

```
for(size_t ic=0; ic<sensVols.size(); ++ic)  {
    PlacedVolume sens_pv = sensVols[ic];
    DetElement comp_elt(module,sens_pv.volume().name(),mod_num);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code start %%%%%%%%%%%%%%%%%%%%%%%%
    Acts::ActsExtension* wafer = new Acts::ActsExtension();
    wafer->addType("passive", "material");
    wafer->addType("axes", "definitions", "XYZ");
    comp_elt.addExtension<Acts::ActsExtension>(wafer);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code end %%%%%%%%%%%%%%%%%%%%%%%%%%
    comp_elt.setPlacement(sens_pv);
}
```

FCCeeCLD

Particularly important:

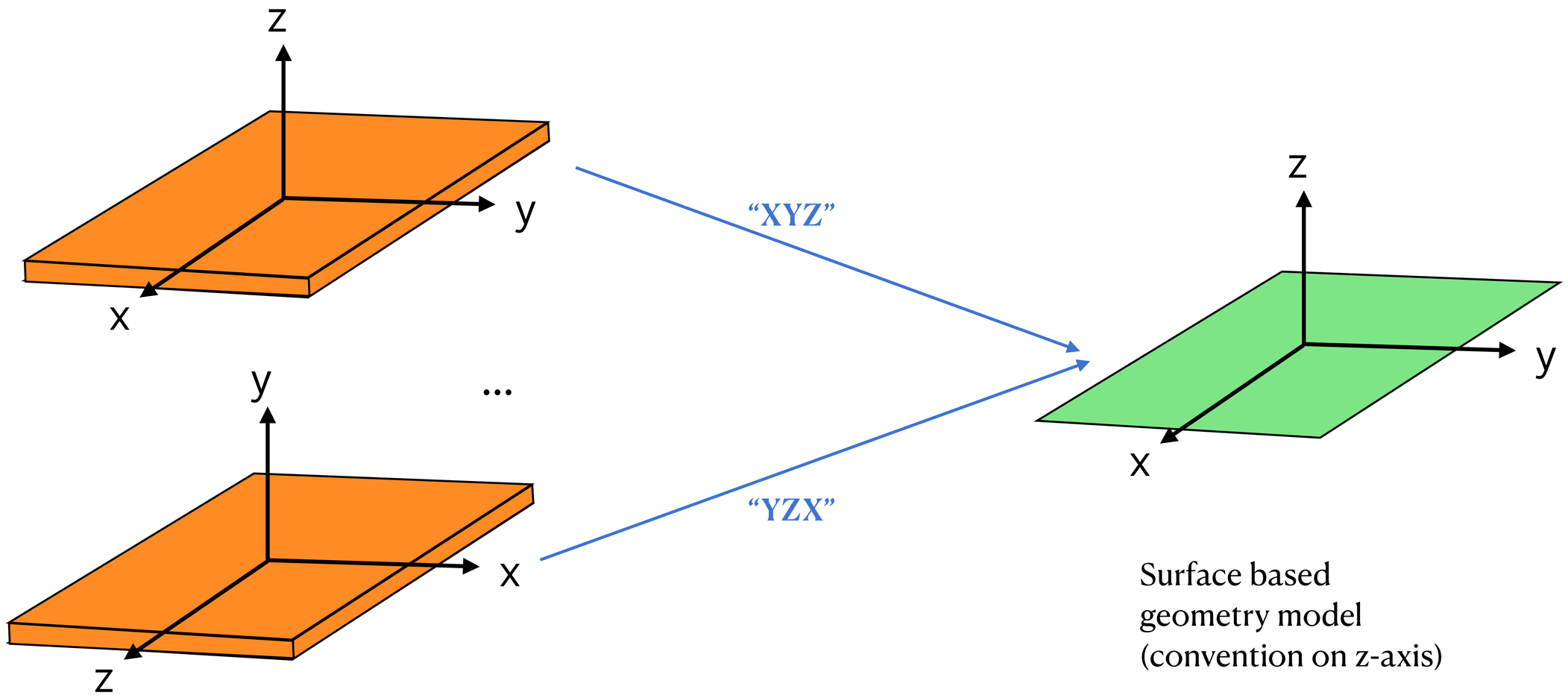
- how does the volume coordinate frame map into the surface coordinate frame ?

```
for(int k=0; k<nmodules; ++k) {
    string m_base = _toString(l_id,"layer%d") + _toString(mod_num,"_module%d") + _toString(k,"_sensor%d");

    double x = -r*std::cos(phi);
    double y = -r*std::sin(phi);
    DetElement module(sdet,m_base+"_pos",det_id);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code start %%%%%%%%%%%%%%%%%%%%%%%%
    Acts::ActsExtension* sensorExtension = new Acts::ActsExtension();
    sensorExtension->addType("sensor", "detector");
    sensorExtension->addType("axes", "definitions", "XZY");
    module.addExtension<Acts::ActsExtension>(sensorExtension);
    /// %%%%%%%%%%%%%%%%%%%% ACTS specific code end %%%%%%%%%%%%%%%%%%%%%%%%%%
```

FCCeeCLD

Axis orientation



3D geometry model

Surface based
geometry model
(convention on z-axis)

Acts::ActsExtension object

```
/// Get the value
///
/// @param tag the entry identifier in the value store
/// @param type the (optional) category in the value store
double getValue(const std::string& tag,
                const std::string& category = "") const noexcept(false);

/// Add the parameter to the store
///
/// @param value the value to be added
/// @param tag the entry identifier in the value store
/// @param type the (optional) category in the value store
void addValue(double value, const std::string& tag,
              const std::string& category = "");

/// Check if the ActsExtension has a value (with optional category)
///
/// @param type the primary identifier in the flag store
/// @param type the (optional) category in the flag store
bool hasValue(const std::string& tag, const std::string& category = "") const;

/// Check if the ActsExtension has a value (with optional category)
///
/// @param type the primary identifier in the flag store
/// @param type the (optional) category in the flag store
bool hasType(const std::string& type, const std::string& category = "") const;

/// Add the characteristics
///
/// @param type the primary identifier in the flag store
/// @param category the (optional) category in the flag store
/// @param the word to be stored
void addType(const std::string& type, const std::string& category = "",
             const std::string& word = "");
```

Nothing more than a string based data base

- is only used for detector construction once hence string based map look up is perfectly fine

Discussion

Is there a way we can communicate these sort of information

- **WITHOUT a dedicated Acts::ActsExtension object**
- **EXTENDABLE?**