Hands on AGDD*:
Atlas Generic Detector Description in XML

Stan Bentvelsen

workshop, Oct 99

*:An initiative from SB & Marc Virchaux

Motivation:

- So far, Atlas did not decide on 'final' database.
- Many applications depend on the Atlas geometry.
- Decision (September): Store Atlas geometry in ascii-files
  - later in data-base
  - Ascii files: easy human readable/editable
- temporary proposal: BAFF format; very flexible + extendible. G4builder available
- Industry standard: XML
  - benefit from all available (free) software
  - well defined standard
  - maintained outside
- XML allows to define syntax yourself.
  - Tailor to specific needs

XML basics

- XML (Extended Mark-up Language). Extension of HTML, ‘tags’ are defined, and xml file comply to these definitions:
  - syntax definition file (dtd-file)
  - implementation file (xml file)
  - Tools available to check the compliance of the xml files (xml4c, xml4j)
- Standard tools to edit, view and parse the XML files
- Standard Atlas software to parse XML in C++ classes (Expat / ExpatInterface)

AGDD basics

Structure from Geant4!

- AGDD: XML definition file to store (generic):
- Solids
  - shape
  - dimensions
  - material
- Boolean solids
  - subtraction, ...
- Positioning of volumes
  - single positioning
  - multiple positioning

- Materials
  - elements
  - composite materials
- Free format information
  - innerstruct

- Additional 'overhead' for administration
  - 'section' element
  - using 'IDREF'
  - Identifiers

provider + client

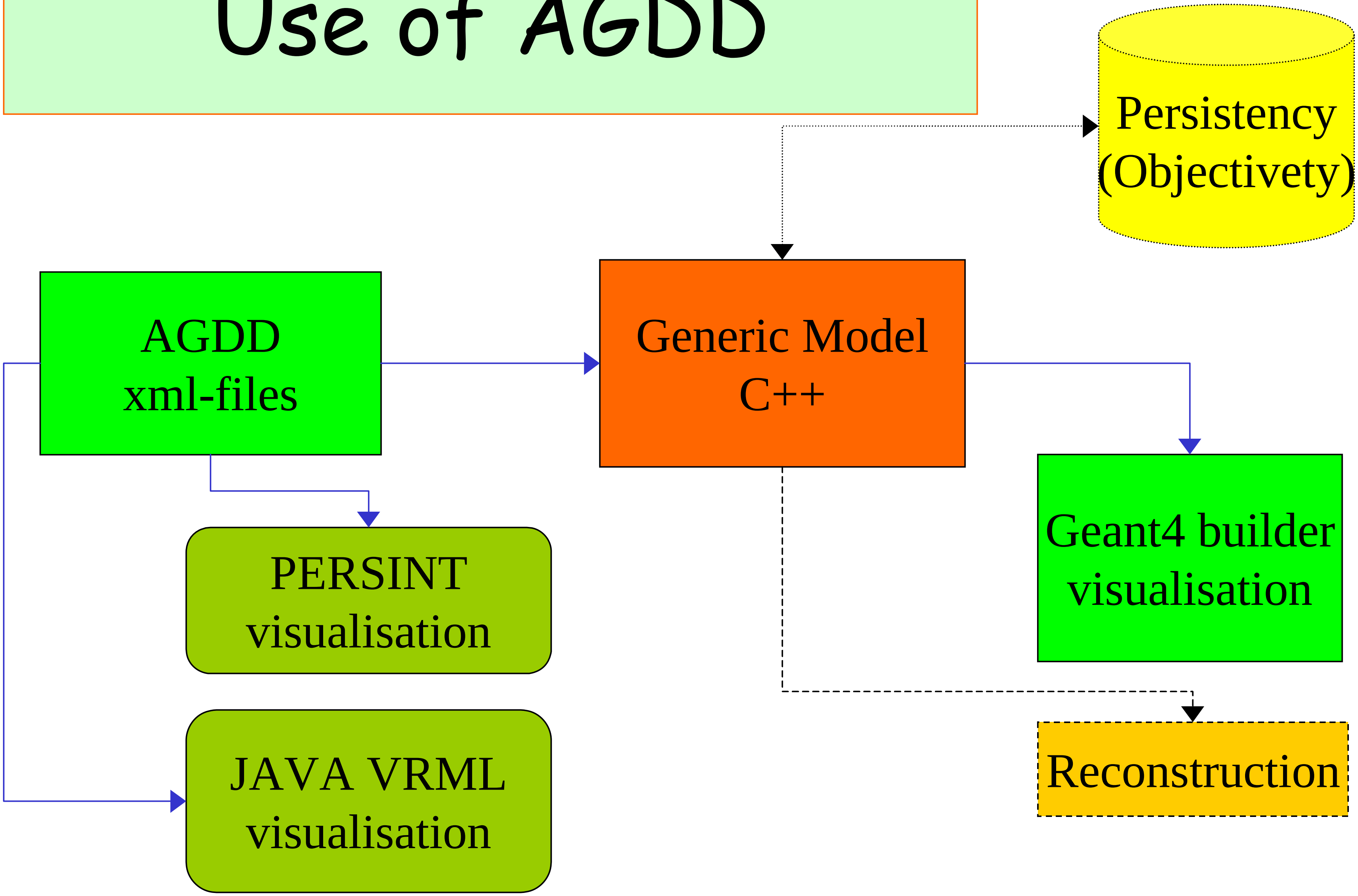
- Provider:
  - AMDB: ascii based database for muon system. Should be easily translated to AGDD
  - Each sub-detector provides a AGDD file with their geometry

  → I.e. you!

- Very low threshold to define your geometry!

- Client
  - Visualisation
    - PERSINT
    - Java, ...
  - Geant 4 simulation package
    - Visualisation
    - Tracking
    - ATLAS simulation
  - Reconstruction
    - Any reconstruction that needs the geometry
  - ....

Use of AGDD



Persistency
(Objectivety)
AGDD
xml-files
Generic Model
C++
PERSINT
visualisation
Geant4 builder
visualisation
JAVA VRML
visualisation
Reconstruction

Quick tutorial AGDD

Since you're here

Solid: definition

- Definition in dtd file

```
<!ELEMENT  solid    EMPTY>
<!ATTLIST  solid
           name        ID       #REQUIRED
           material    IDREF    #REQUIRED
         shape       ( box | trd | tubs )    #REQUIRED
           dim         CDATA    #REQUIRED
           innerstruct IDREF    #IMPLIED
           %units;
>
```

  - 'Solid' element is empty, it has only attributes
  - attribute list define the solid completely
  - #REQUIRED: obligatory
  - #IMPLIED: not obligatory, no default
  - ID: defines XML identifier
  - IDREF: reference to a XML identifier
  - %units: refer to predefined entity 'units' (compare to 'alias')

Solid: implementation

- Three equivalent ways to define the same solid in a implementation AGDD.xml file

```
<solid name="SCT_wafer" shape="box" material="Silicon" dim="0.3 63.6 64" />

<solid dim="0.3 63.6 64" material="Silicon" name="SCT_wafer" shape="box" />

<solid name="SCT_wafer"
     shape="box"
     material="Silicon"
     dim="0.3 63.6 64"
 /solid>
```

- XML parser reads these formats
- attribute values free format!
  - E.g. no checking for 4 real numbers in ‘dim’
  - Need to create an additional tool for strong type-checking

SCT module

- Position a board and an electronics board to create a module:

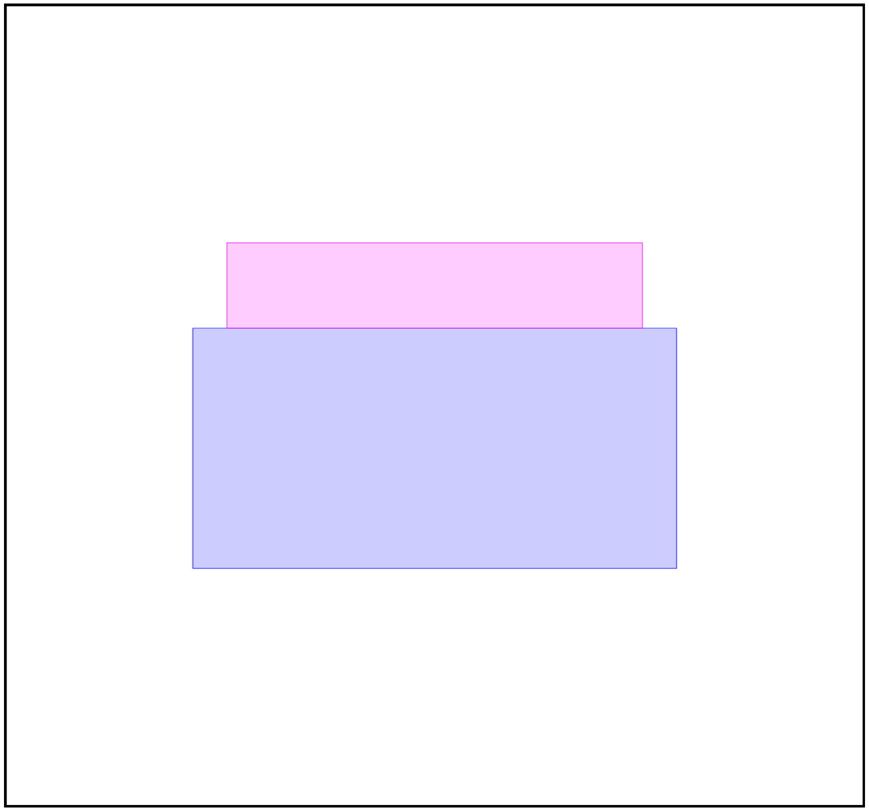
XML comment

```
<!-- position wafer+hybrid board -->
<composition name="SCT_module">
   <posXYZ  volume="SCT_board"    X_Y_Z= " 0   11.3   0"     />
   <posXYZ  volume="SCT_elboard"  X_Y_Z= " 0 -31.8   0"     />
</composition>
```

- <composition> has elements inside that are positioned. The coordinate system is defined by the resulting volume.
- No rotation in this example
- the ‘SCT_module’ itself can be positioned

Module

- Interactive display
  - In this example, I've used PERSINT
- Nothing much to say: it's a very simple module
  - not a very realistic piece!



Composition definition

- Definition of a composition: placing volumes

```
<!-- Definition of a composition:
     The envelope attribute points to a volume
 -->
<!ELEMENT  composition     (posXYZ|posRPhiZ|mposR|mposPhi|mposZ)+>
<!ATTLIST  composition
           name      ID       #REQUIRED
           envelope  IDREF    #IMPLIED
>


<!--
  posXYZ   : Positioning of a single volume in Carthesian coordinates.
  ======     The volume can be rotated before it is placed.
             The coordinate system is defined by the placement itself.
-->

<!ELEMENT  posXYZ EMPTY>
<!ATTLIST  posXYZ
           volume    IDREF     #REQUIRED
           X_Y_Z     CDATA     "0 0 0"
           rot       CDATA     "0 0 0"
           %index;
           %units;
>
```

- posXYZ is single placement: translation + rotation

A step back: Volumes

- Volume: generic name for a geometric object
- ‘Solid’ and ‘Composition’ are both ‘volumes’
- 5 volume-types:
  - Solid
  - Composition
  - Boolean volumes:
    - Union
    - Intersection
    - Subtraction

- Volume has
  - name
  - units
  - Solids:
    - shape
    - dimensions
    - material
    - (innerstruct)
  - Composition
    - list of positionings
  - Boolean volumes
    - reference volume
    - list of positionings

Positionings

- Placement of a volume inside another volume
  - thereby creating the 'mother volume'
- 'Mother' volume defined as the union of all its daughters
  - no explicit shape or dimensions!
  - Envelope may, but does not need, to be given.

- Single positioning
  - posXYZ
  - posRPhi
- Multiple positioning
  - mposPhi
    - 'ncopy' daughter volumes along phi, fixed (R,Z)
  - mposR
    - along R, fixed (phi,Z)
  - mposZ
    - along Z, fixed (R,phi)

Loose ends

- Unit system: default to “mm” and “deg”

```
<!ENTITY % units 'unit_length  (mm|m)         "mm"
                  unit_angle   (deg|mrad)     "deg"'>
```

- Innerstruct:
  - Each volume may point to a innerstruct. This is a reference to a free fromat to store detector specific information (strip-pitch, nr of channels/crystal etc..)
- Index: (first version: needs to be iterated? Better?)
  - system to identify each solid
  - useful when solid is placed many times
  - currently foresee 3-dimensional indexing (I_r, I_phi, I_z)

```
<!ENTITY % index 'index      CDATA "0 0 0"'    >
```

SCT ski

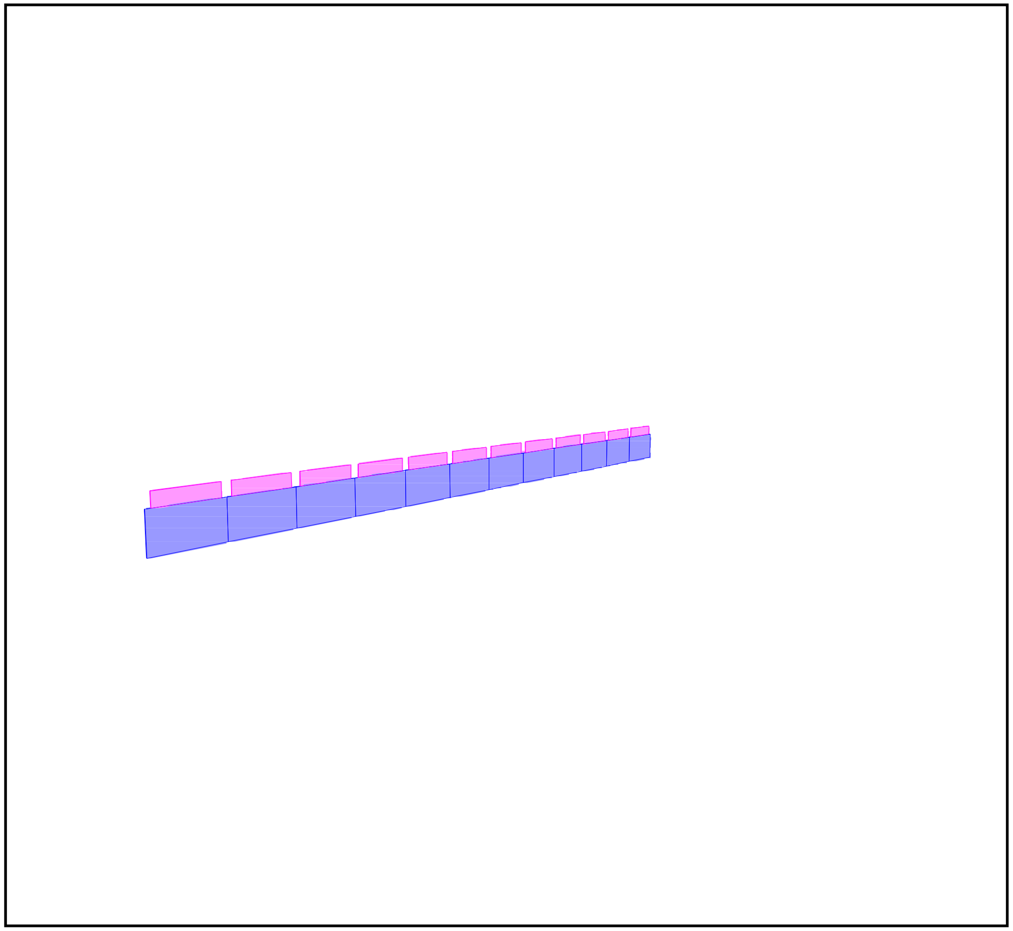
- Create a ski as a ‘mposZ’ of 12 modules:

```
<!-- create a ski of 12 modules (Ir=1 to 12) -->
<composition name="SCT_ski">
   <mposZ   volume="SCT_module"  ncopy="12"  dZ="128.2"  Z0="-769.2" index="0 0 1"  />
</composition>
```

- Give n-copy, Z-start, Z-step
- Index: “0 0 1”
  - I_r=0 ; I_phi = 0 ; I_Z=(1-12)
  - Each copy gets its own index: “0 0 1” towards “0 0 12”
  - First two 0’s not used
  - Index-step default to 1.

Ski: visualisation

- Visualisation with PERSINT



SCT barrel

- Create SCT_barrel: 4 ‘mposPhi’ elements, with 32, 40, 48 and 56 copies, respectively
- Rotation needed!

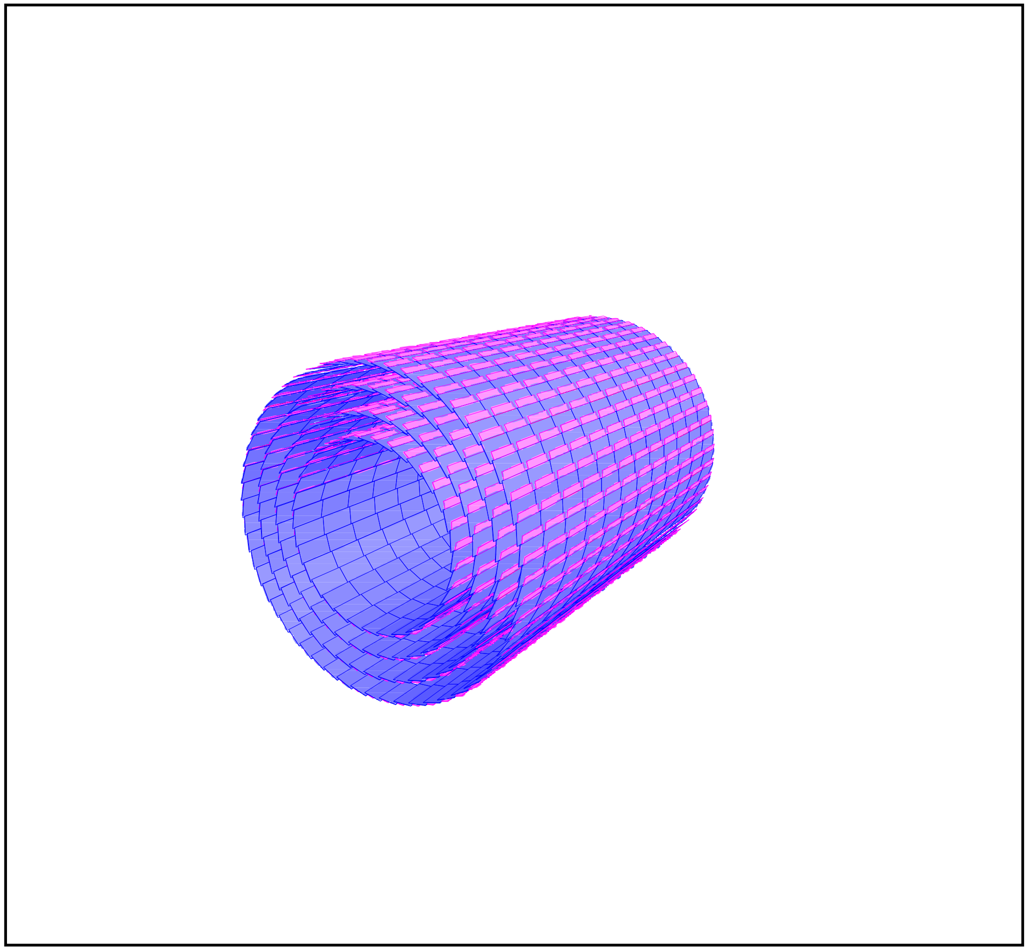
```
<!-- make the barrel sct consisting of four (Ir=1 to 4)  rings (Iphi=1 to n)
 of skis of 4 modules (Iz=1 to 4), each with a tilt of 10 degrees            -->

<composition name="SCT_Barrel">
   <mposPhi volume="SCT_ski"  ncopy="32" R_Z="300.0 0"  rot="0 0 -10"  index="1 1 0"  />
   <mposPhi volume="SCT_ski"  ncopy="40" R_Z="373.0 0"  rot="0 0 -10"  index="2 1 0"  />
   <mposPhi volume="SCT_ski"  ncopy="48" R_Z="447.0 0"  rot="0 0 -10"  index="3 1 0"  />
   <mposPhi volume="SCT_ski"  ncopy="56" R_Z="520.0 0"  rot="0 0 -10"  index="4 1 0"  />
</composition>
```

- Index: “1 (1-n_{copy}) 0” to “4 (1-n_{copy}) 0” for ring 1-4.
  - I_Z remain (1,12) from ski definition
  - I_phi runs between (1,ncopy) for each ‘mposPhi’
  - I_R given explicit for each ring (larger R value)

Barrel: visualisation

- Visualisation using PERSINT



Sections

- Place a geometry inside a 'section' (all attributes are required)

```
<section name       = "SCT"
         version    = "1.1"
         date       = "Thu Oct 7"
         author     = "Stan Bentvelsen"
         top_volume = "SCT_Barrel"   >
```

- Name
  - Short descriptive name; each volume is prepend with this name (to resolv name-space problem in XML)
- Version, date, author
  - To keep track of the geometry version
- Top_volume
  - Envelope volume name corresponding to the section

Putting it together

- Extremely easy-to-make geometry files
- Direct visualisation with various tools (persint, JAVA, G4builder)

```
<?xml version="1.0"?>
<!DOCTYPE AGDD SYSTEM "AGDD_1.04.dtd">

<AGDD>

<!--  Atlas Generic Detector Description : test for SCT
     ***************************************************
-->
<section name       = "SCT"
         version    = "1.1"
         date       = "Thu Oct 7"
         author     = "Stan Bentvelsen"
         top_volume = "SCT_Barrel"  >


<!-- create a wafer+hybrid board -->
<solid  name="SCT_board  "  material="Silicon"  shape="box"  dim="1. 63.6 128.2"
/>
<solid  name="SCT_elboard"  material="Copper"   shape="box"  dim="1. 22.6 110.0"
/>

<!-- position wafer+hybrid board -->
<composition name="SCT_module">
   <posXYZ  volume="SCT_board"    X_Y_Z= " 0  11.3  0"    />
   <posXYZ  volume="SCT_elboard"  X_Y_Z= " 0 -31.8  0"    />
</composition>

<!-- create a ski of 4 modules (Ir=1 to 12) -->
<composition name="SCT_ski">
   <mposZ   volume="SCT_module"  ncopy="4"  dZ="128.2"  Z0="-769.2"    index="0
0 1"  />
</composition>

<!-- make the barrel sct consisting of four (Ir=1 to 4)  rings (Iphi=1 to n)
 of skis of 4 modules (Iz=1 to 4), each with a tilt of 10 degrees             --
>
<composition name="SCT_Barrel">
   <mposPhi volume="SCT_ski"  ncopy="32"  R_Z="300.0 0"  rot="0 0 -10"  index="1
1 0"  />
   <mposPhi volume="SCT_ski"  ncopy="40"  R_Z="373.0 0"  rot="0 0 -10"  index="2
1 0"  />
   <mposPhi volume="SCT_ski"  ncopy="48"  R_Z="447.0 0"  rot="0 0 -10"  index="3
1 0"  />
   <mposPhi volume="SCT_ski"  ncopy="56"  R_Z="520.0 0"  rot="0 0 -10"  index="4
1 0"  />
</composition>

</section>
```

Materials

- Separate XML tags for definition of materials

```
<materials version  ="1.1"
           date     ="Thu Oct 7"
         author   ="Stan Bentvelsen" >

<!-- Define the elements -->

  <element name="Hydrogen    " symbol="H "   z=" 1"   aweight="1.00797" />
  <element name="Helium      " symbol="He"   z=" 2"   aweight="4.0026"  />
  <element name="Lithium     " symbol="Li"   z=" 3"   aweight="6.941 "  />
  <element name="Beryllium   " symbol="Be"   z=" 4"   aweight="9.0122"  />
  <element name="Boron       " symbol="B "   z=" 5"   aweight="10.81 "  />
  <element name="Carbon      " symbol="C "   z=" 6"   aweight="12.011 " />
  <element name="Nitrogen    " symbol="N "   z=" 7"   aweight="14.0067 "/>

<!-- ...etc....-->

  <composite  name="Scintillator" density="1.032">
    <addmaterial material="Carbon"     fraction="9"  />
    <addmaterial material="Hydrogen"   fraction="10" />
  </composite>


</materials>
```

- Whole idea stolen from del'Acqua's G4 material-manager (but implementation needs iterations!)

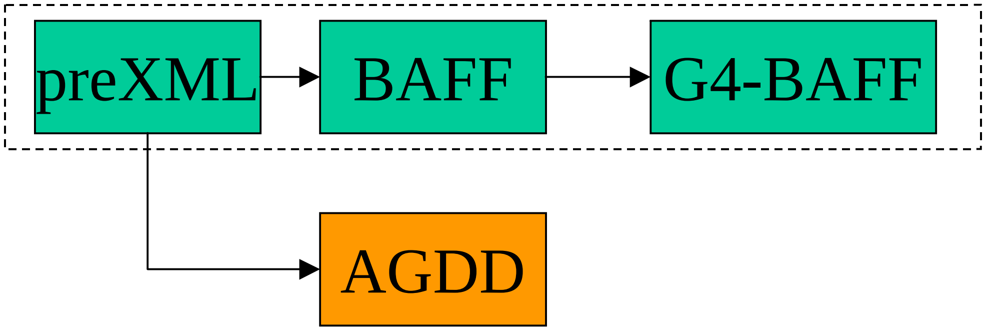
ToDo list

- Divide the ATLAS detector in components.
  - Each sub-detector has its own 'envelope'
  - Define 'envelopes' in separate file
  - refer to those envelopes in the 'section'
  - Each sub-detector can fill their envelope as they please
- Who is volunteering for this?

- 'Translate' the AMDB database to this AGDD XML format
  - automatic translation?
- Couple detector construction database to AGDD?
  - Simple ascii output; should be possible
  - direct link construction vs simulation vs reconstruction

The story of the barrel SCT

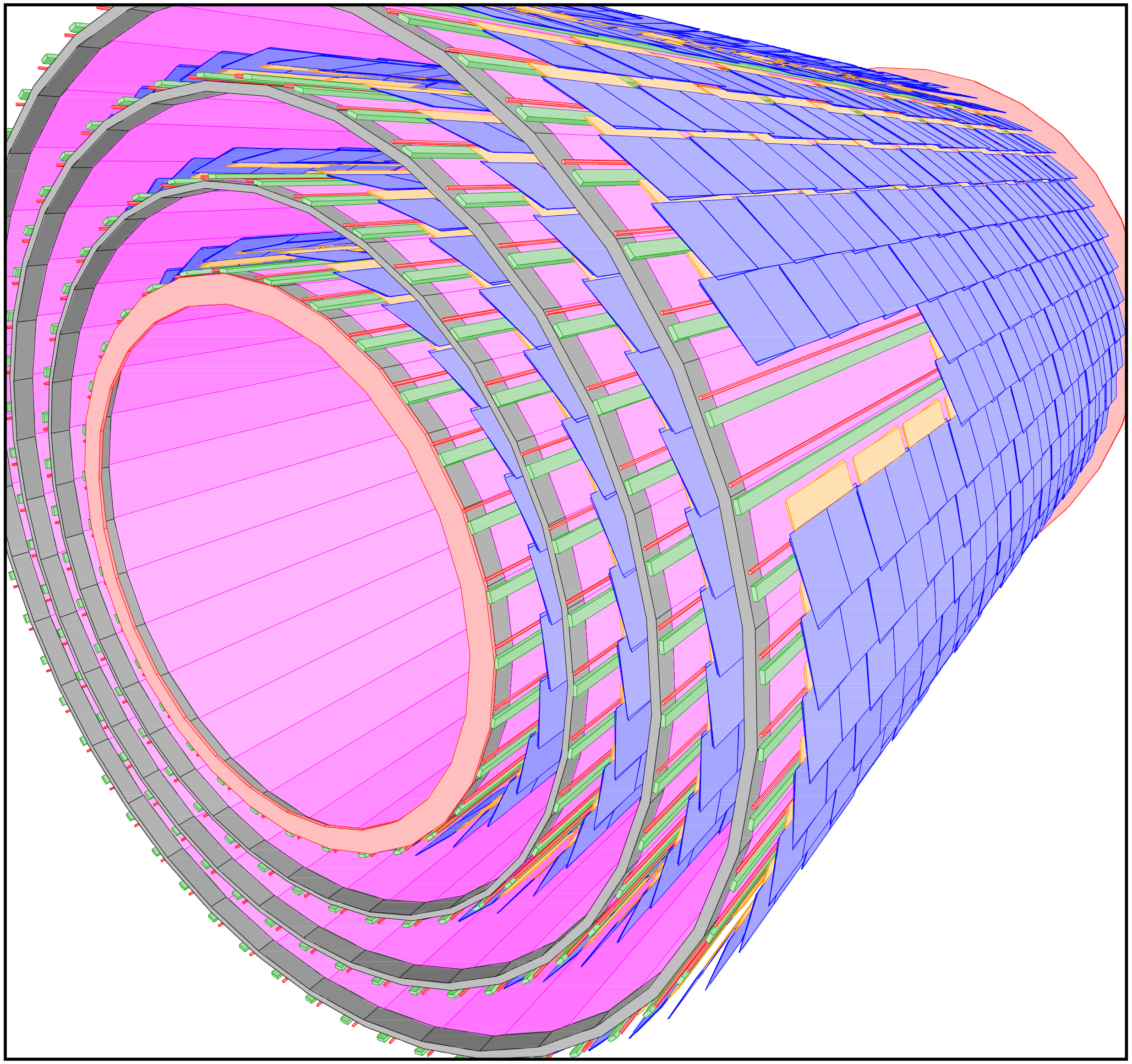
- AGDD is only few weeks old.
  - Before that, there was BAFF + G4-BAFF builder
- Christopher Lester created SCT barrel DICE geometry in BAFF format



- He noticed that many numbers in XML file depend on each other
  - create his 'own', private XML syntax
  - no dependencies in this preXML.
  - create a small C++ program that output either BAFF or AGDD format
  - He handed me a AGDD file that he couldn't visualise with AGDD himself (no G4builder yet!)

SCT

- DICE geometry
- AGDD file provided by Lester
- PERSINT visualisation
- Who follows??



A Geant4 builder

Stan Bentvelsen
workshop, Oct 99

Why a G4-builder

- The whole AGDD syntax is rather close to Geant-4
- Geant-4 simulation is one of the 'important' clients of XML
- Build a 'generic' detector independent Geant-4 builder
- Clearly not the 'end of the story'

- First attempt to create a generic G4 builder
- Use Geant-4 visualisation: 'DAWN'

- No attempt to track particles yet!

G4builder

Del’Acqua algorithm

- Interface to C++ Generic Model
- pick-up the ‘visit_solid’ and ‘visit_composition’ so far
- ‘Root’ volume ‘ATLAS’ is visualised
- No (re-) compilation when XML files are changed!

- Problem of mother-volumes:
  - in XML no specific mother volume defined
  - when a new volume is positioned in mother:
    - create a union of the current mother and the new volume
    - position mother in the union
    - position volume in the union
    - call union the ‘new mother’

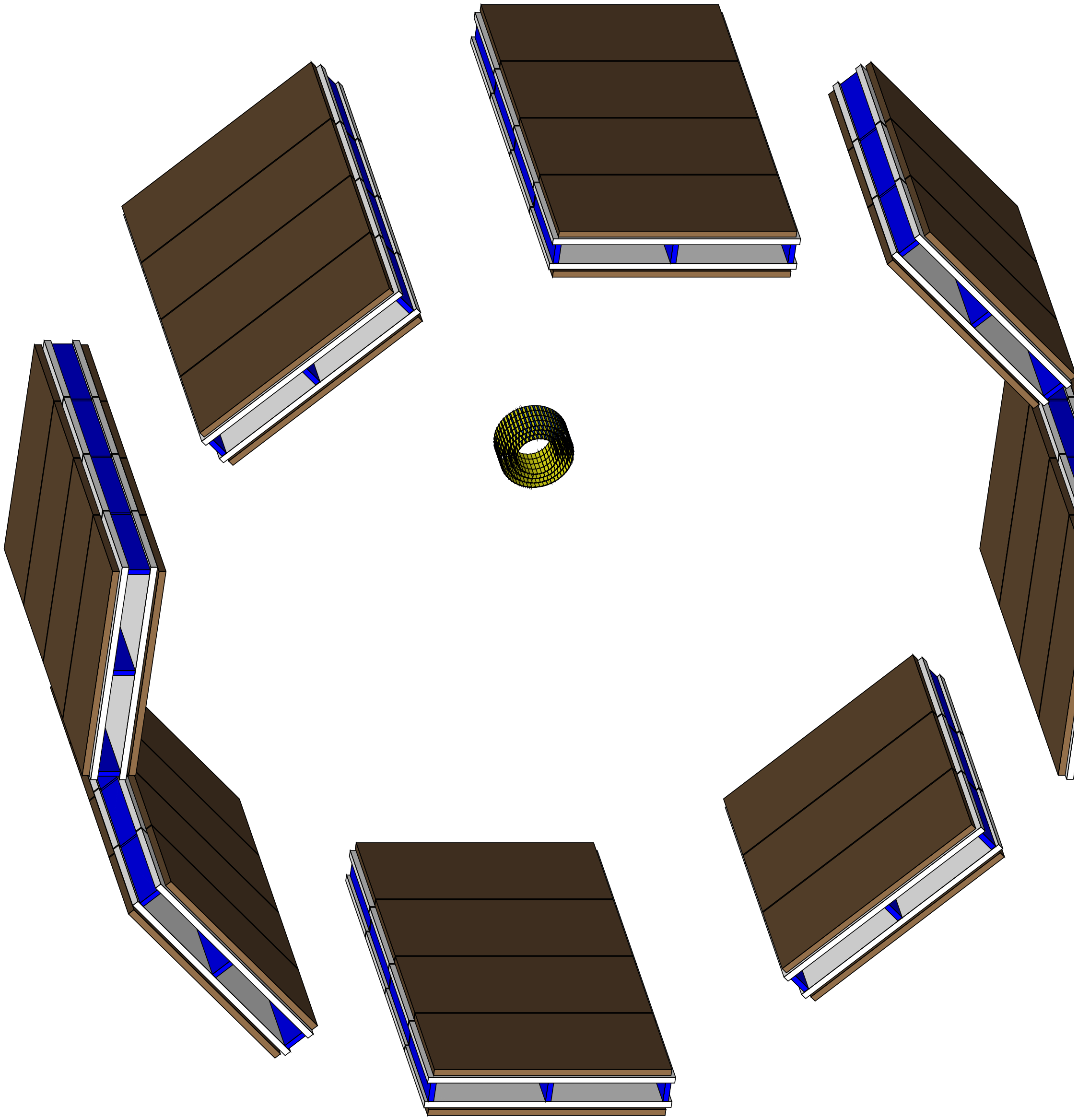
G4builder

- Materials:
  - Interface between 'visit_material' and MaterialManager (A. del'Acqua)
  - All elements available, composite materials need to be worked on
  - Do not know all the details for 'materials'; need to be discussed further

- Status:
  - Current version extremely preliminary
    - written in a couple of days only
    - it works
  - use private version of Geant-4 (with STL)
  - 'Linux' version of Geant4 has problem:
    - need to modify a g++ include file
    - not possible on afs?

Test

- Geant-4 display of the Test_AGDD xml file



DICE

- Christopher Lester
  SCT barrel geometry
- Show ring 4 only