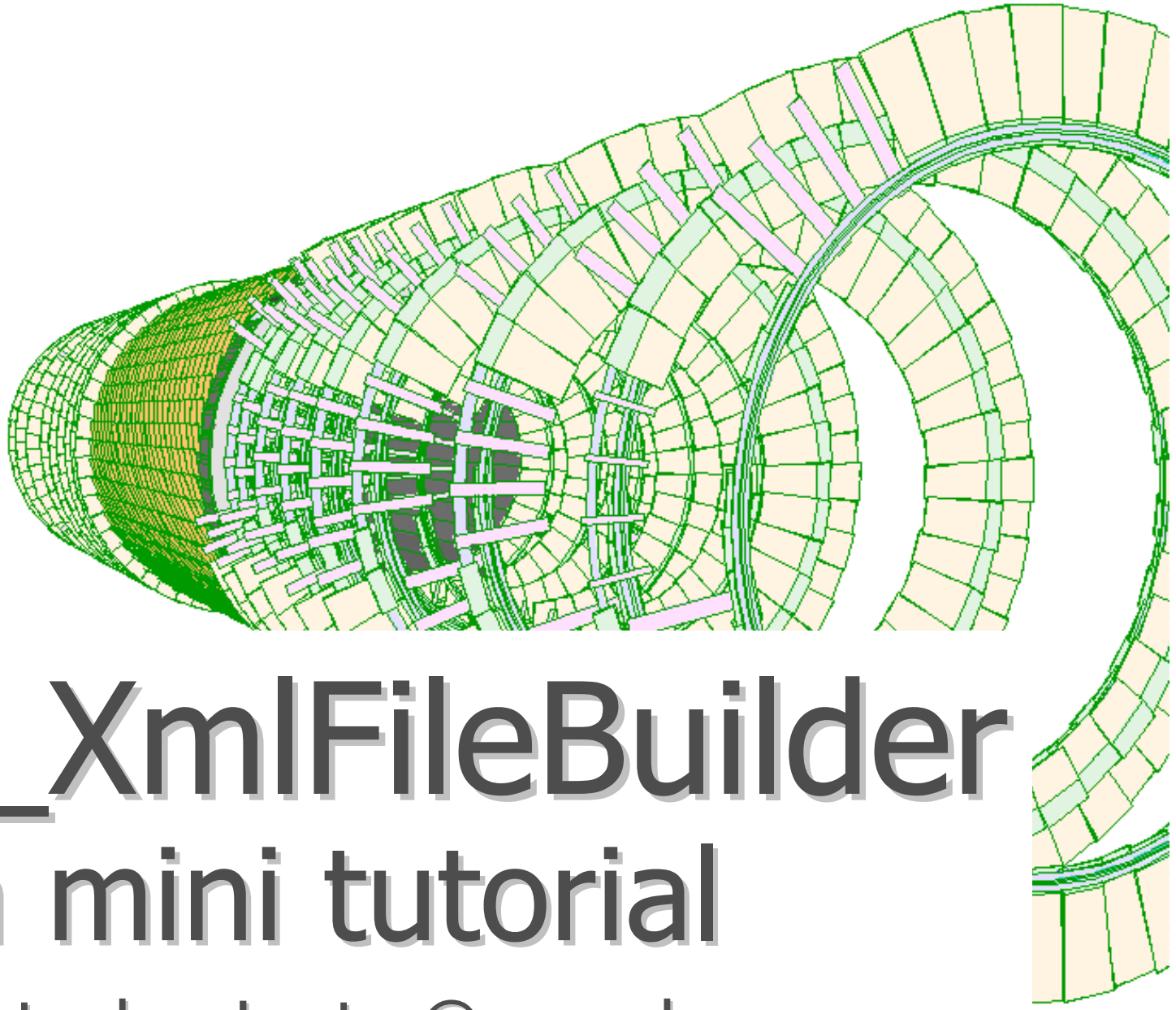


AGDD_XmlFileBuilder
- a mini tutorial

Christopher.Lester@cern.ch

What is AGDD_XmlFileBuilder?

- It's a set of C++ tools which allow easy creation and management of complicated AGDD/XML files
- The set of C++ tools comprises
  - library
  - user tool classes
  - (and implementation classes …)

Where is AGDD_XmlFileBuilder?

- It is in the ATLAS software area, under offline/DetectorDescription/AGDD_XmlFileBuilder
- Some "real" use by the SCT community can be found in: offline/InnerDetector/InDetDetDescr/SCT_AGDD_XmlFileBuilder, although these may not be the best examples to learn from.
- Simple examples of use may be found in the following pages ….

What AGDD features does AGDD_XmlFileBuilder support?

Primary solids

- Box, Tubs, Cons, Trd, HalfTrd*

Boolean solids

- Union, Intersection, Subtraction

Positioners / Identifiers

- PosXYZ, MPosPhi, Identifier (inc. Layer/Ring/Sector forms)

Administrative

- Compos, Section, AGDD

If what you want is not here, campaign for it!

From C++ to XML

This is a three stage process:

1: Main stage: Define your detector description (boxes, composes etc) by using the tools.

2: "Build" the description.

3: Output the XML.

Example : Box

```
#include "AGDD_XmlFileBuilder/Box.hh"

// Make new box:

agdd::Box * myBox = new agdd::Box;


// Set it's parameters:

myBox->setName("myWafer");

myBox->setBox(length, width, height);

myBox->setMaterial("silicon");    // various possibilities ...
```

Example : PosXYZ

```
#include "AGDD_XmlFileBuilder/PosXYZ.hh"

// Make a positioner. We will use PosXYZ:

 agdd::PosXYZ * boxPos = new agdd::PosXYZ;


// Set the positioner's parameters:

boxPos->setVolume(myBox);          // Says what to position

boxPos->setRot(10*degree, 0, 0);   // Gives a rotation

boxPos->setXYZ(0, 100*mm, 0);      // Gives a translation
```

Example : Compos

```
#include "AGDD_XmlFileBuilder/Compos.hh"

// Make a composition:

 agdd::Compos * doubleWafer = new agdd::Compos;


// Fill the compostion:

doubleWafer->setName("double_wafer");  // Name the compos

doubleWafer->addPos(boxPos);                // What to put in

doubleWafer->addPos(someOtherBoxPos); // … etc ...
```

Example : Section

```
#include "AGDD_XmlFileBuilder/Section.hh"

// Make a section:

 agdd::Section * mySection = new agdd::Section;


// Set parameters ...

mySection->setName("SCT_barrel_section");

mySection->setTopVolume(doubleWafer);

mySection->setAuthor("ChristopherLester");

mySection->setVersion("1.2.3");
```

Example : AGDD

```
#include "AGDD_XmlFileBuilder/AGDD.hh"


// Make an AGDD in memory:
 agdd::AGDD * myAGDD = new agdd::AGDD;


// Set top volume, build geometry, and produce XML output!
myAGDD->addSection(mySection1);
myAGDD->addSection(mySection2);          // … etc ...
```

Building 1/3 : RootBuildables

A RootBuildable is a component of AGDD/XML whose output is supported by the tools.

At the moment, Section and AGDD are the only RootBuildables supported by the tools, so these are the smallest elements which may be "built" and output.

Building 2/3 : buildTheRoot()

While the detector is being "described" by the user, no internal representations are being constructed.

When the user believes his description is complete, he must make one call to the buildTheRoot() method of his topmost RootBuildable. This completes the internal building process.

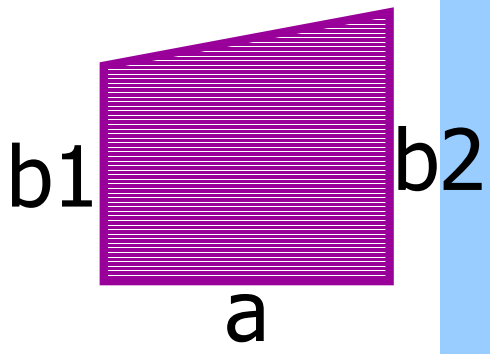
Building 3/3 ... and output !

```
#include "AGDD_XmlFileBuilder/Section.hh"


// Suppose we have a section which we want to print ...
 agdd::Section * mySection = new agdd::Section;
/* ... section definition omitted ... */


mySection->buildTheRoot();   // Completes internal build.
mySection->sendMeTo(cout);  // Prints to stream.
mySection->sendMeTo(ofstream("out.xml"));  // Prints to file.
```

Advanced features

Use shapes which are not in the DTD, without forcing Marc to produce a new version of Persint!

```
#include "AGDD_XmlFileBuilder/HalfTrd.hh"

// Make a halfTrd:

agdd::HalfTrd * shapeNotInDTD = new agdd::HalfTrd;


// Set the top volume, and print XML file!

shapeNotInDTD->setName("wedge");

shapeNotInDTD->setHalfTrd(a, b1, b2, thickness);
```



Advanced features

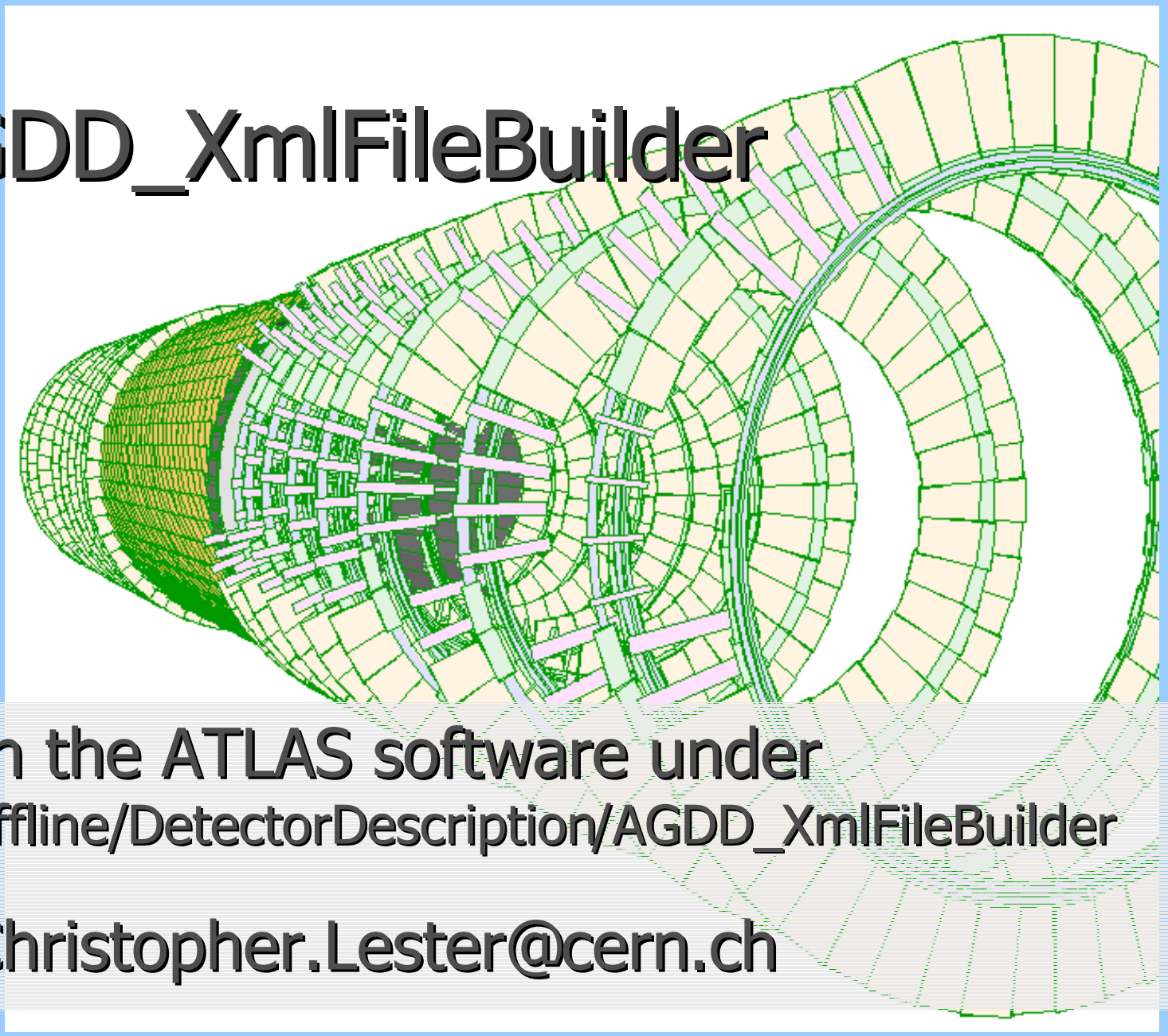
When two things are similar but not quite the same ...

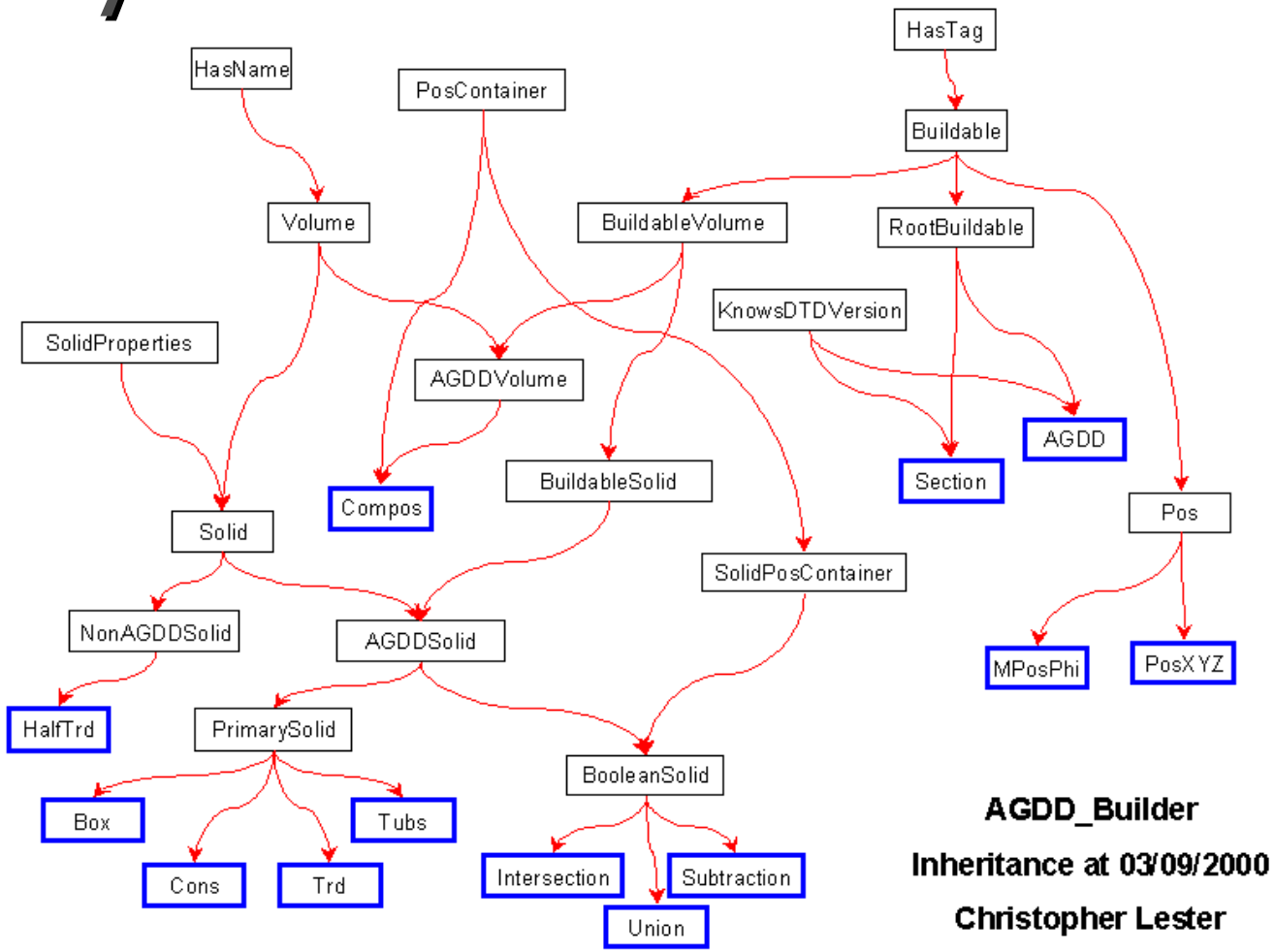
```
// A function to return a complicated object:
agdd::Compos * makeComplicatedLayer(int layerNumber) {
        agdd::Compos * layer = new agdd::Compos;

        layer->setName("SCT_layer_"+layerNumber);
        if (layerNumber==2) {
                // Build the layer in a special way
        } else {  /* Build the layer in the normal way */  };

        return layer;
};
```

Remember!

- Tools must update with each DTD
  - Hopefully no changes needed in end user's geometry code if update to DTD is minor!
  - Possibly changes will be forced if change is major!
- Three things you must bear in mind when versioning:
  - What was your input data?
  - Which version of the tools did you use?
  - What version of your implementation code did you use with the tools and the input data?

AGDD_XmlFileBuilder



- In the ATLAS software under
  offline/DetectorDescription/AGDD_XmlFileBuilder
- Christopher.Lester@cern.ch

Internal workings ...
only for the brave!