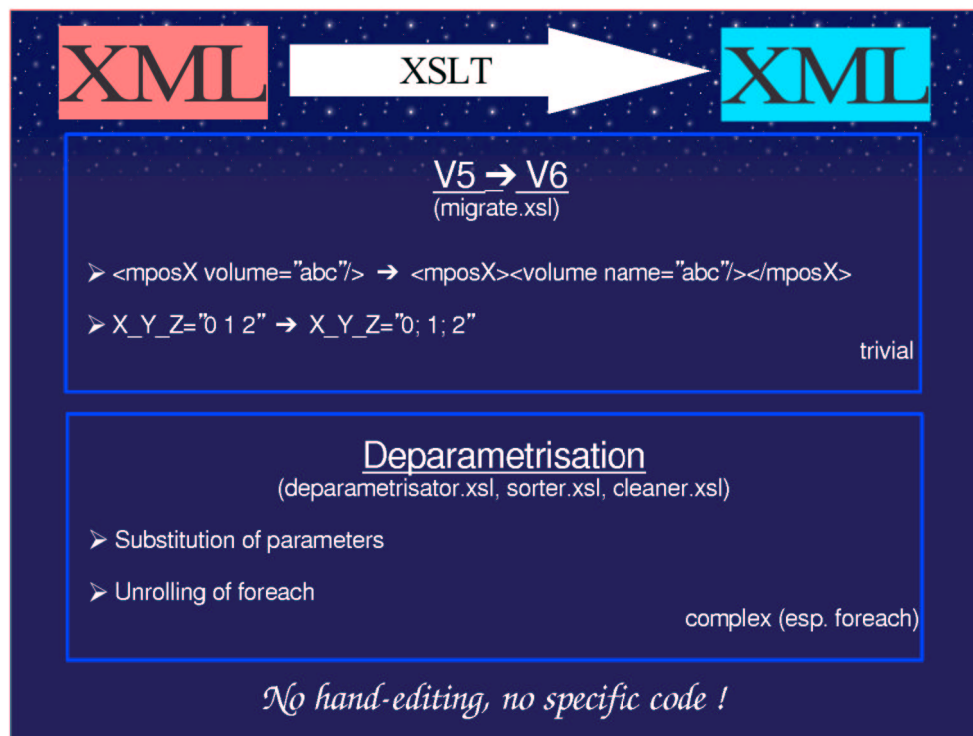
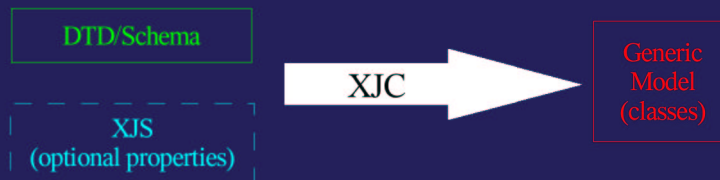




- XSLT Stylesheets allow for automatic translation of XML files to anything else (another XML files, documentation, source code,,,) without writing any additional processing code.
- The call is 'xalan -in in.xml -out out.xml -xsl transformation.xsl' (in case of Xalan XSLT processor - there are others too).
- CMT contains fragment for XSL processing.
- Two (sets of) XSLT stylesheets have been created.
- V5->V6: Translation of current AGDD into proposed one. This is very simple, it just changes handling of 'volume's and format of vectors (blank -> ;). It is proof of the claim (made before) that choosing XML will make any future migration easier.
- Deparametrisation: Changes AGDD with all new (proposed) features (variables, foreach loops) into standard AGDD. This is the same operation as done by C++ classes in case of Compact elements; the difference is that XSLT is general, while those classes exist one for each Compact element. Other difference is, that XSLT can be used standalone, while those special C++ class should be run within a Framework. Deparametrisation stylesheet is quite complex.

JAXB Generic Model

- Third way of XML processing, besides SAX and DOM.
- Automatic creation of processing code from formal specification.
- Similar to ADL concept.
- Generated classes are not general ("Element"), but specific ("StackX") and they have specific methods ("StackX.getOrigin()").



```

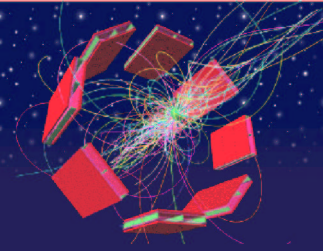
<!-- Definition of common interface for several classes. -->
<interface name="Solid" members="box trd" properties="unit_length unit_angle"/>

<!-- Definition of automatic conversion -->
<attribute name="sensitive" convert="boolean"/>

<!-- Definition of conversion provided by service class -->
<attribute name="X_Y_Z" convert="DoubleArrayListCnv"/>
  
```

- There are three ways of XML processing:
 - SAX – sequential
 - DOM - generic Tree of 'Elements'
 - JAXB - binded Tree of special objects, source code created automatically
- The creation of classes which represent XML elements (and processing code) is completely automatic. One can write a special configuration file, which customises the creation for things like:
 - Grouping of elements to common interfaces (by inheritance).
 - Defining simple types (double, int,...), which are converted automatically from strings in XML into proper types.
 - Defining more complex types (like vector of doubles), which are converted by simple conversion classes (which have to be written by hand).
- Many more...
- The procedure works well with any flavor of our AGDD XML.

GraXML Generic Model



- GraXML contains complete Generic Model (inherited from Java3D), which is used to perform graphical tasks:
 - Drawing (calculating of transformation matrices)
 - Picking (following volumes along rays)
 - Navigation (association of volumes with identifiers)
 - ...
- This Generic Model can be:
 - Extracted from GraXML
 - Stripped off graphical properties (colors,...)
 - Merged with JAXB Generic Model
 - Interfaced via standard API (once we have it) to all supported languages (Java, C++, F**)

- GraXML (as any other graphics program) has its internal Generic Model.
- This Generic Model does already many tasks, which are quite similar to tasks required from AGDD Generic Model:
- Other similar functionalities can be added with the help of Java3D.
- Those tasks can be easily customised to work not only for graphics. One has to remove all graphics-specific elements too (like colors, interactivity,...).
- This Generic Model can quite easily work with many different flavors of detector description XML files (already now) - not only from Atlas.
- It is useless to have just a bunch of objects in memory, we have to define clear API. Once this happens, service in any language (like Java, C++, Fortran) can serve clients in other languages.