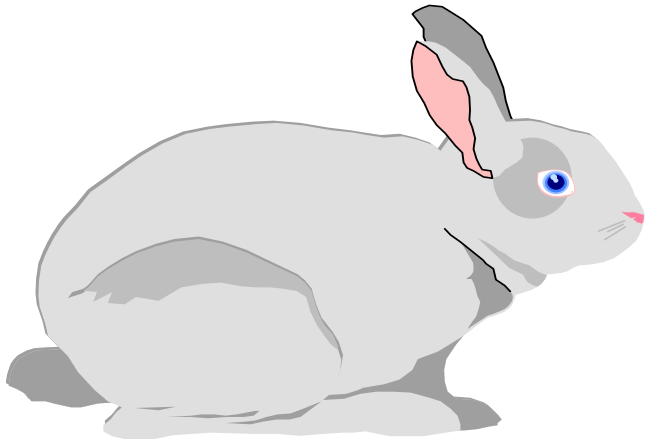
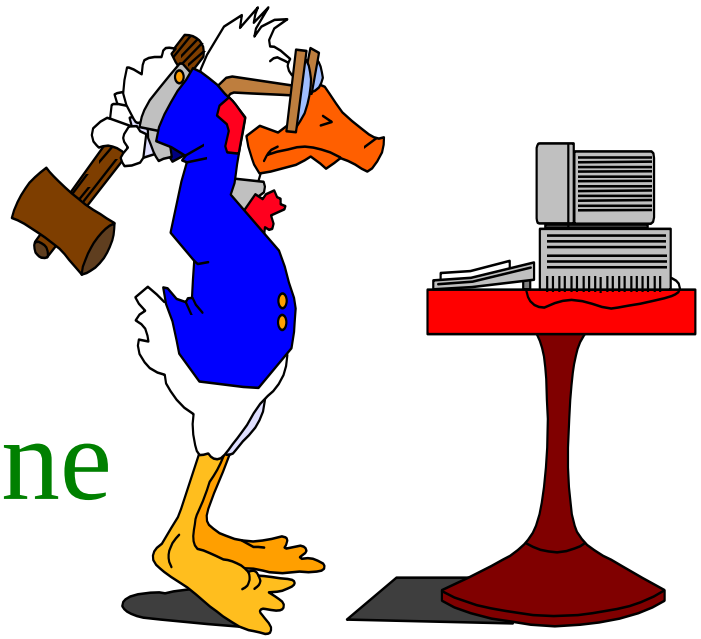
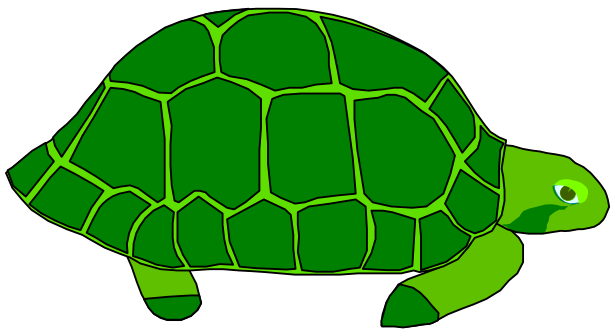




Identifiers for Everyone



J. Bogart

Core Software Workshop

April 16-20 2001

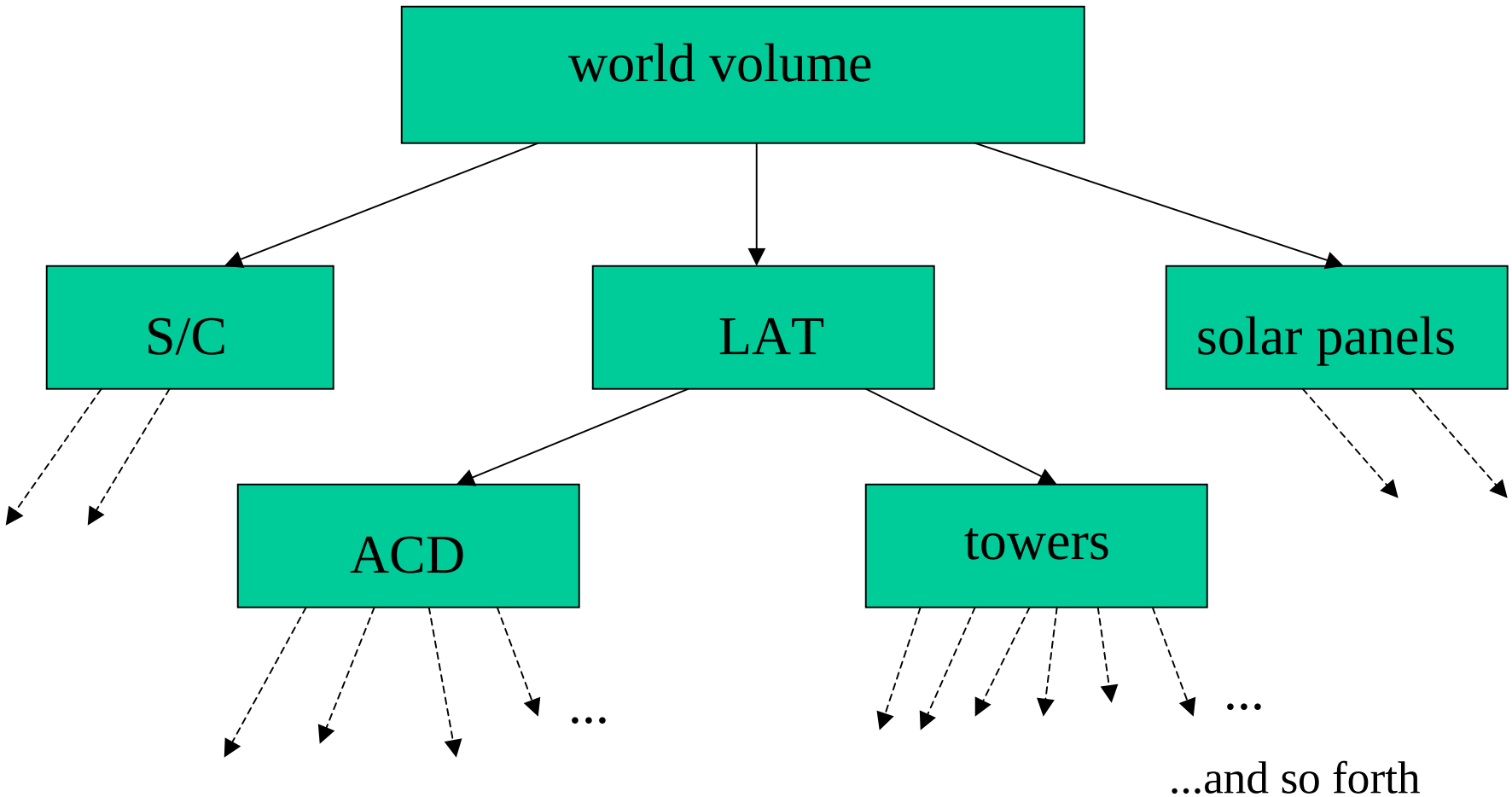
detModel Volume Hierarchy

The GDD.. classes within the detModel package make an in-memory representation of the information in (for example) flight.xml. It keeps track of volume shapes, materials, and how they are positioned with repect to each other and a global coordinate system.

Complex volumes are formed by positioning existing volumes w.r.t. an envelope. One volume may wholly contain another,but otherwise they may not intersect.

This leads to a natural hierarchy of volumes or tree with a single root, the “world volume.”

GLAST GDD Volume Hierarchy



Traversing the Volume Tree

The world volume contains the LAT

which contains a tower #3

which contains a CAL

which contains layer #2

which contains a CsI crystal #4

It would be natural to associate an identifier which is a sequence of fields, like /0/3/1/2/4, with this object.

Assigning Id Fields

Detector geometry XML allows assignment of one or more idFields (a name and a value) each time a volume is positioned.

Form complete **GDD Identifier** for a volume by concatenating idfields of all ancestor volumes, starting with (implicit) world volume.

Only certain sequences will form valid GDD identifiers (there is no layer #27). Use more XML in the form of an **id dictionary** to keep track of this.

Why We're Not Done

There are constraints imposed by simulators on exactly how the volumes can be put together. Also the xml syntax we're using provides shortcuts for certain kinds of positioning.

As a result the nesting described in flight.xml can be less than ideal for other purposes.

Why We're Not Done (examples)

- Towers get two numbers: (row, column)
- Components like the grid have to be artificially broken up into per-tower pieces, acquiring extra irrelevant tower numbers
- A tracker "layer" (closest X and Y) can't be assigned a single identifier since the Si belongs to 2 different trays.

Three Kinds of Identifiers

- GDD identifiers as just described
- Readout identifiers, used for digis
- Volume identifiers, used to identify hits (both integrating and position).

One Approach

- Although GDD identifiers aren't always ideal for a particular application, they can be constructed to contain all the information any client might want
- Readout and volume identifiers can be given a similar form (sequence of field names with values), so their constraints may also be defined by an id dictionary.
- Can describe transformations needed to get from
  - GDD identifier to Volume id
  - GDD identifier to Readout id

One Approach (cont'd)

- Generic identifier dictionary classes and identifier conversion classes can do much of the work.
- Classes specific to the kind of identifier (GDD identifier, volume id, etc.) will be needed for some services, such as look-up.
- Check against use cases to be sure everything we need is covered!

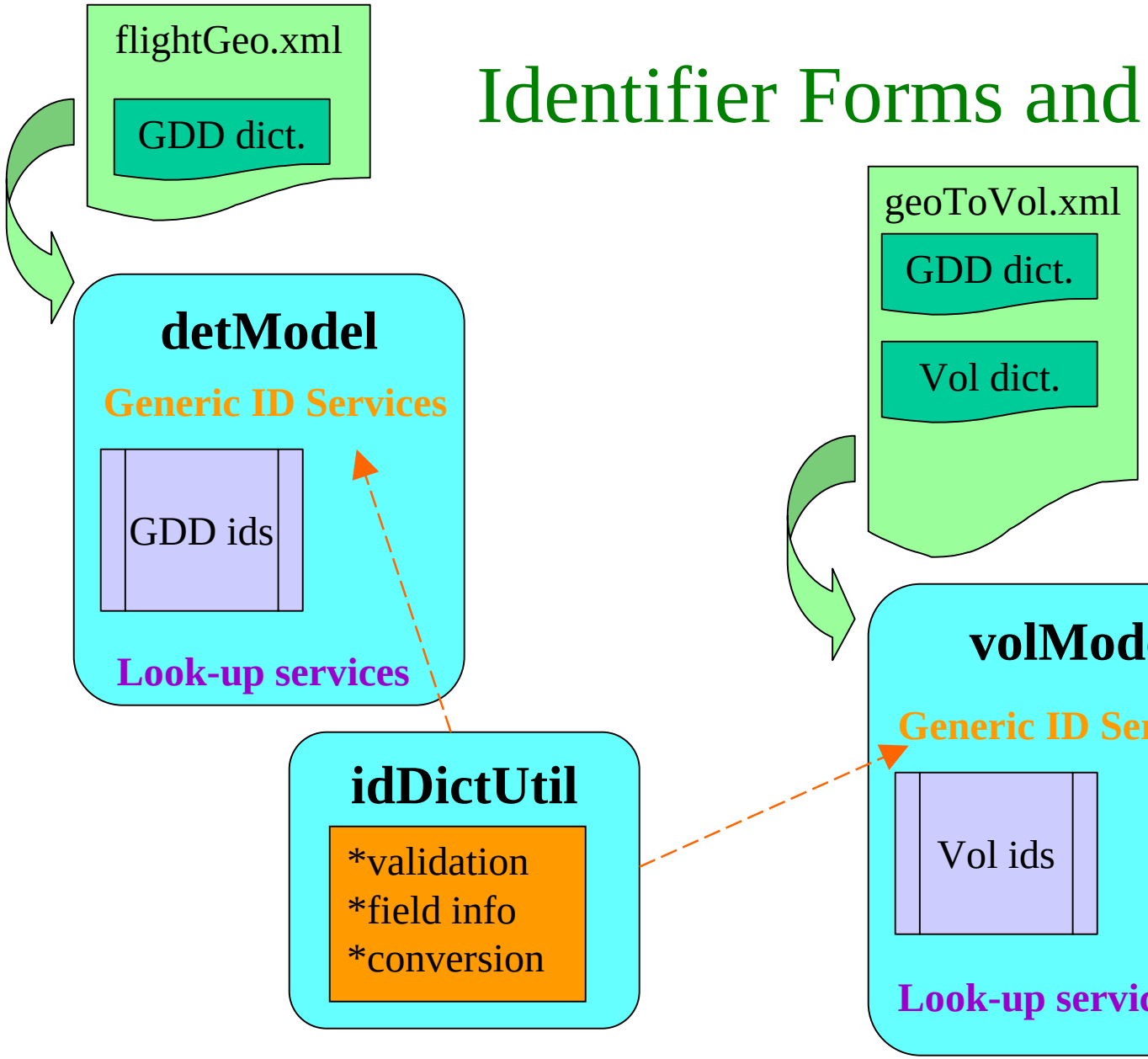
Simulation

- detModel uses flight.xml to build the geometry. GDD ids are supplied to simulator to label volumes
- During simulation as hits are instantiated GDD ids are converted to volume ids, which are used to label the hits.
- If possible do something similar (translate GDD ids to readout ids) for digis.

Event Data Client

- detModel uses flight.xml to build the geometry.
- Invoke a detModel visitor whose output is a data structure associating volume ids with the “original” volumes in the detModel.
- The new data structure supports look-up operations by volume id.
- Do the same thing for readout ids.

Identifier Forms and Services



detModel visitor (not shown) uses geoToVol. xml to create volModel, which provides look-up via volume id for, e.g. Reconstruction.