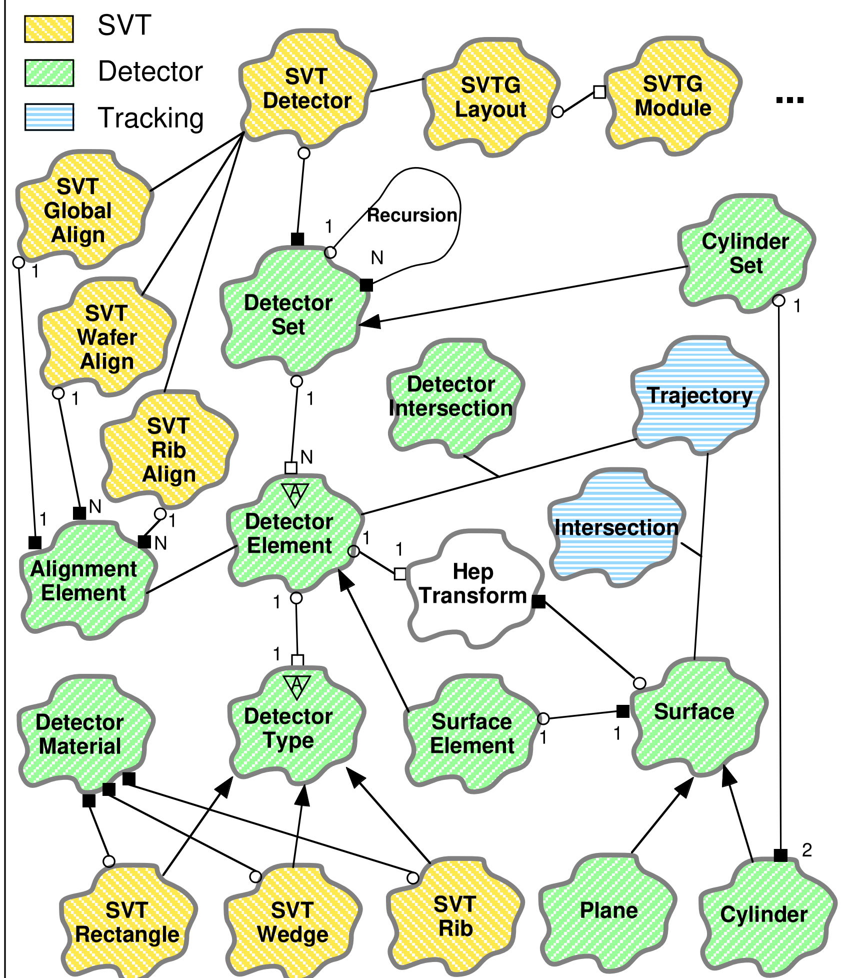
DetectorModel Package

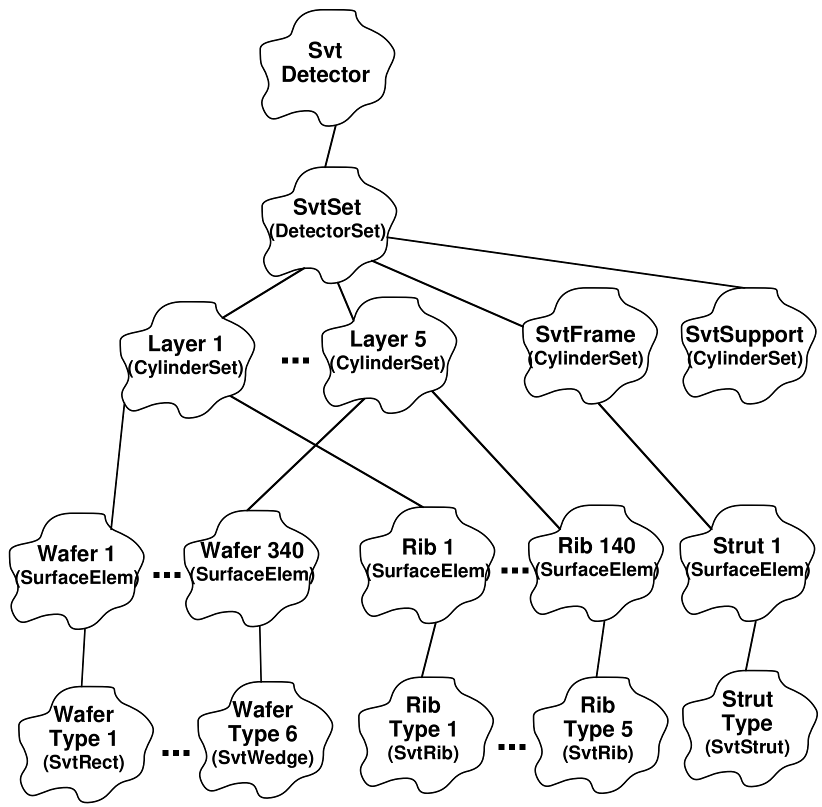
The first release of the DetectorModel package will be made tomorrow. This package provides base classes for describing the geometry and material of HEP detectors. It provides interfaces and services for tracking, nominal geometry, alignment, and material effects. This package will be used by the SVT, DCH, and DIRC to describe their detectors. This presentation provides a brief overview.

- DetectorModel Classes
- SvtTrack Package Objects
- Geomety and Material Interface
- Tracking Interface
- Alignment Classes and Interface
- Possible ProxyDict AbsEnv Database Interface

DetectorModel + SVT Classes



SvtTrack Objects



Geometry and Material

- The external geometric description of an element uses HepTransforms (CLHEP)
  - Every DetectorElement references a transform.
  - Transforms describe the position and orientation of objects in the BaBar coordinate system.
  - DetectorSets do not have intrinsic geometric properties (IE element transforms are free standing).
  - The transform also translates space points into ‘local’ points (DetectorCoordinate) and vice-versa.
  - Surfaces (planes and cylinders) are implemented using transforms.
  - The same transform is used to describe passive (material) and active (hits) aspects of an element.
  - Alignment methods act on this transform.
- The internal geometric description of an element can be 2-D or 3-D
  - Interface (DetectorType) is mostly virtual.
- Material properties are described by DetectorMaterial helper class.
  - Multiple scattering, energy loss, etc.
  - Collaboration with Geant4 on these is sensible

Tracking Interface

- The tracking interface is through Trajectories
  - A Trajectory defines a 1-D object in 3-D space.
    - A Track must provide a trajectory.
    - A hit must provide a trajectory.
    - Residuals are defined as the distance between 2 trajectories.
  - A DetectorElement provides methods to determine if and where a (track) trajectory intersects it.
    - DetectorIntersection describes entrance, exit, and effective interaction point as path lengths along the trajectory.
  - A DetectorElement can describe the material effects (multiple scattering, energy loss, etc.) corresponding to a DetectorIntersection.
  - DetectorSet provides navigation methods.
    - Next intersection (next DetectorElement intersected).
    - All intersections (within a specified path length range).
    - Daughter classes (CylinderSet) are designed to do this very efficiently: Navigating the SVT (~500 elements) takes ~ 15msec*.
- Active elements provide methods to translate a measurement into a hit trajectory.
  - Reconstruction details are hidden, as measurements are described in TypeCoordinates.
  - DetectorType objects might be shared with ‘clustering’.
- Kalman tracking uses the material interface
  - Used in KalMaterial constructor and functions.

* on a DEC ALPHA 250 MHz- model V

Alignment Interface

- DetectorModel implements alignment using the AlignElement class.
  - Parameters represent small deviations from nominal.
  - A HepTransform can be constructed from these.
  - Alignments can be *applied* and *removed.*
    - Apply sets the element transform to the product of the existing transform and the alignment transform. T *= A.
    - Remove does the same for the inverse alignment transform.
  - Alignment can be applied *globally* or *locally* .
    - global alignment multiplies the element transform on the left side.
    - local alignment multiplies the element transform on the right side. The DetectorElement and AlignElement names and IDs must match exactly.
  - Both global and local alignment can be applied through DetectorSet.
    - Global applies a single alignment to all contained elements.
    - Local uses an array of alignments, one for each DetectorElement.
- The aligned position of a DetectorElement is implicit in its transform.
  - Hit trajectories and material navigation are ‘automatically corrected’ for alignment with no overhead.
- No alignment bookkeeping is kept internally.
  - Avoids inefficient checks (efficient checks are outside).
  - Avoids problems when computing alignment.

AbsEnv Interface

- The existing AbsEnv will be replaced with a proxy dictionary.
- SvtDetector is the only proxied object for the (SVT) DetectorModel.
    - The fault handler builds the SvtDetector object if none already exists.
        - Nominal geometry/material and alignment objects are collected from the database by the fault handler.
        - These are passed to the SvtDetector constructor as pointers not OORefs.
        - The fault handler keeps a pointer to the SvtDetector object.
        - The SvtDetector is set to 'expire' upon the next event.
    - The fault handler updates the SvtDetector object if it already exists.
        - All database objects are tested to see if they have changed since the last event. If not, the fault handler is reset to expire on the next event and the existing SvtDetector pointer is returned.
        - If any database objects (alignment) changed, the SvtDetector is modified.
            - Global alignment changed -> remove old alignment and apply new.
            - Local alignment changed -> delete and rebuild SvtDetector object (?)
    - The SvtDetector class does not know about Objectivity.
        - Only the proxy fault handler has to deal with the database.

Conclusions

- The DetectorModel package is part of the next release.
    - A partial implementation of the Svt built on this is also being released (SvtTrack)
    - An implementation of the Dirc is imminent.
- The package is built on existing core software.
    - CLHEP
    - RW template libraries.
- The package provides a standard interface to tracking.
    - Material navigation.
    - Hits.
- The package provides a standard implementation of alignment.
- Collaboration with Geant4 on a material description class is desireable.
    - A working version is provided.