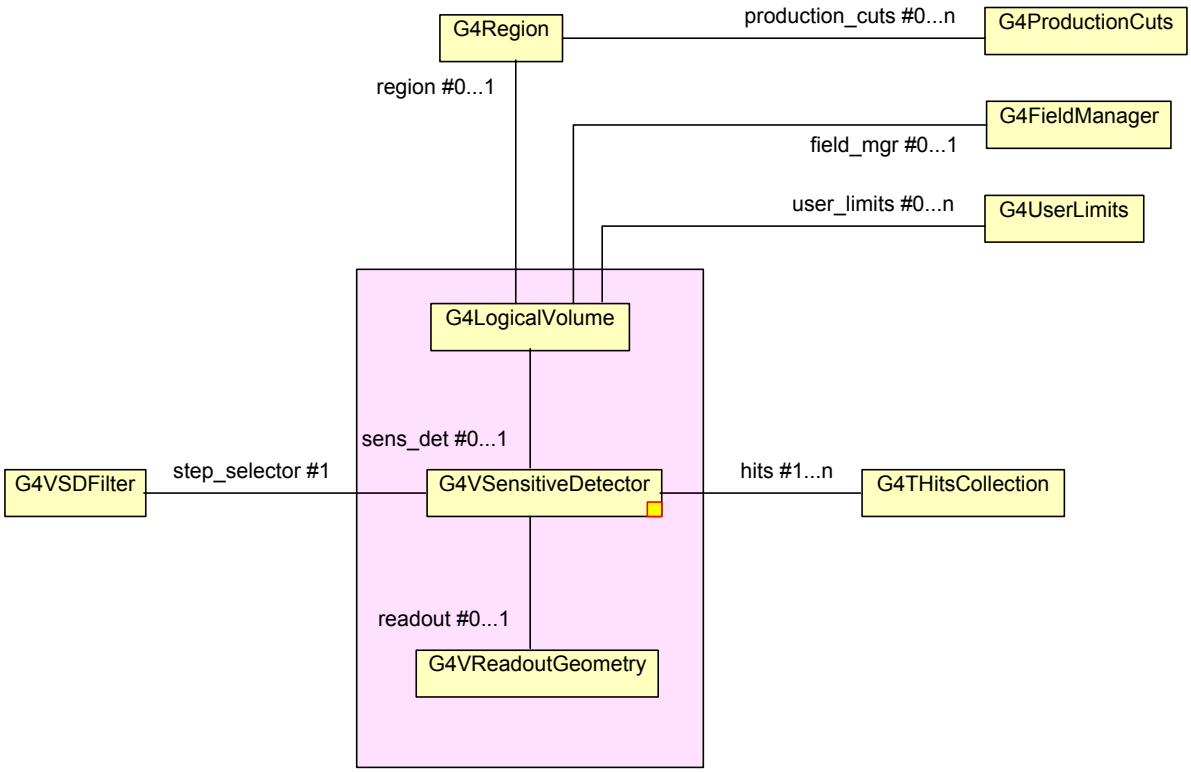
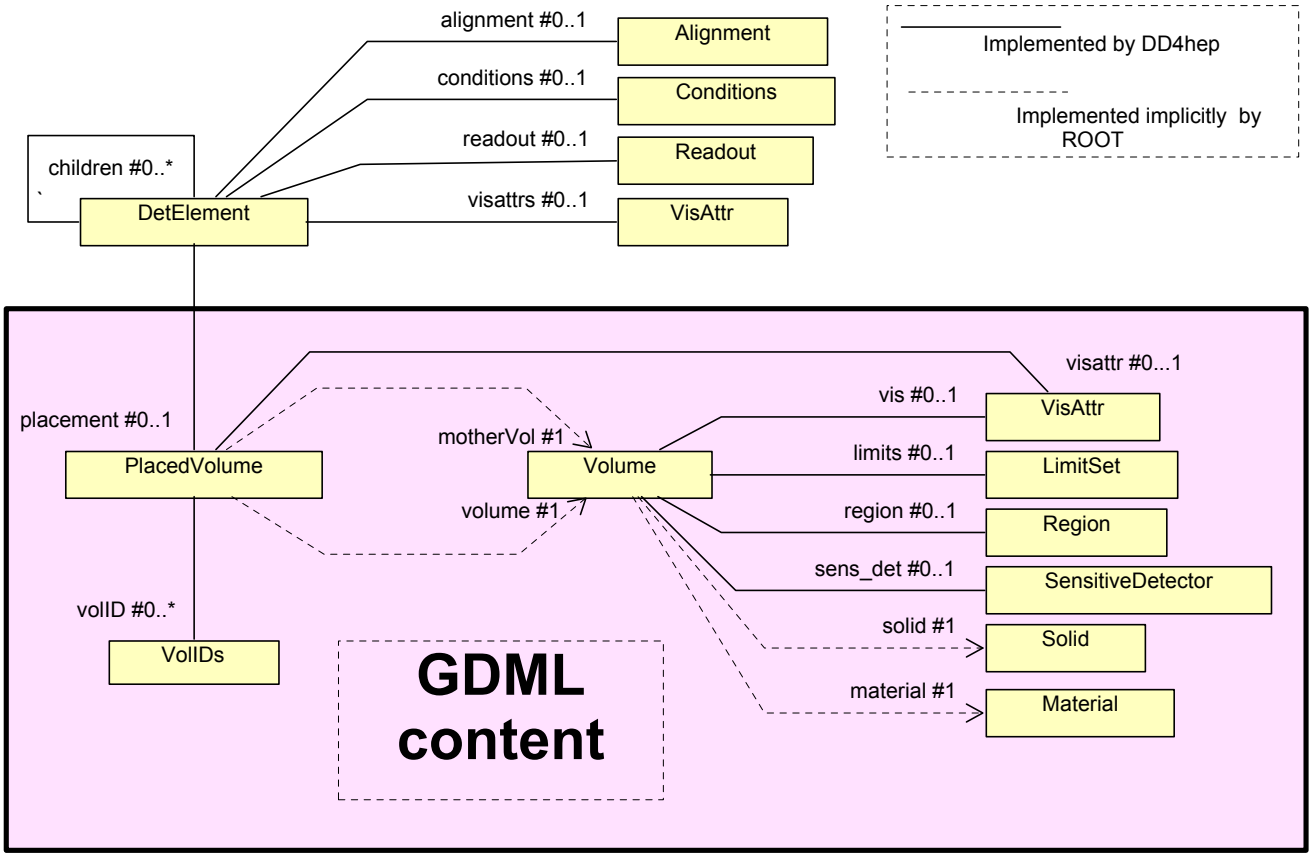
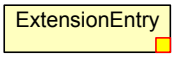
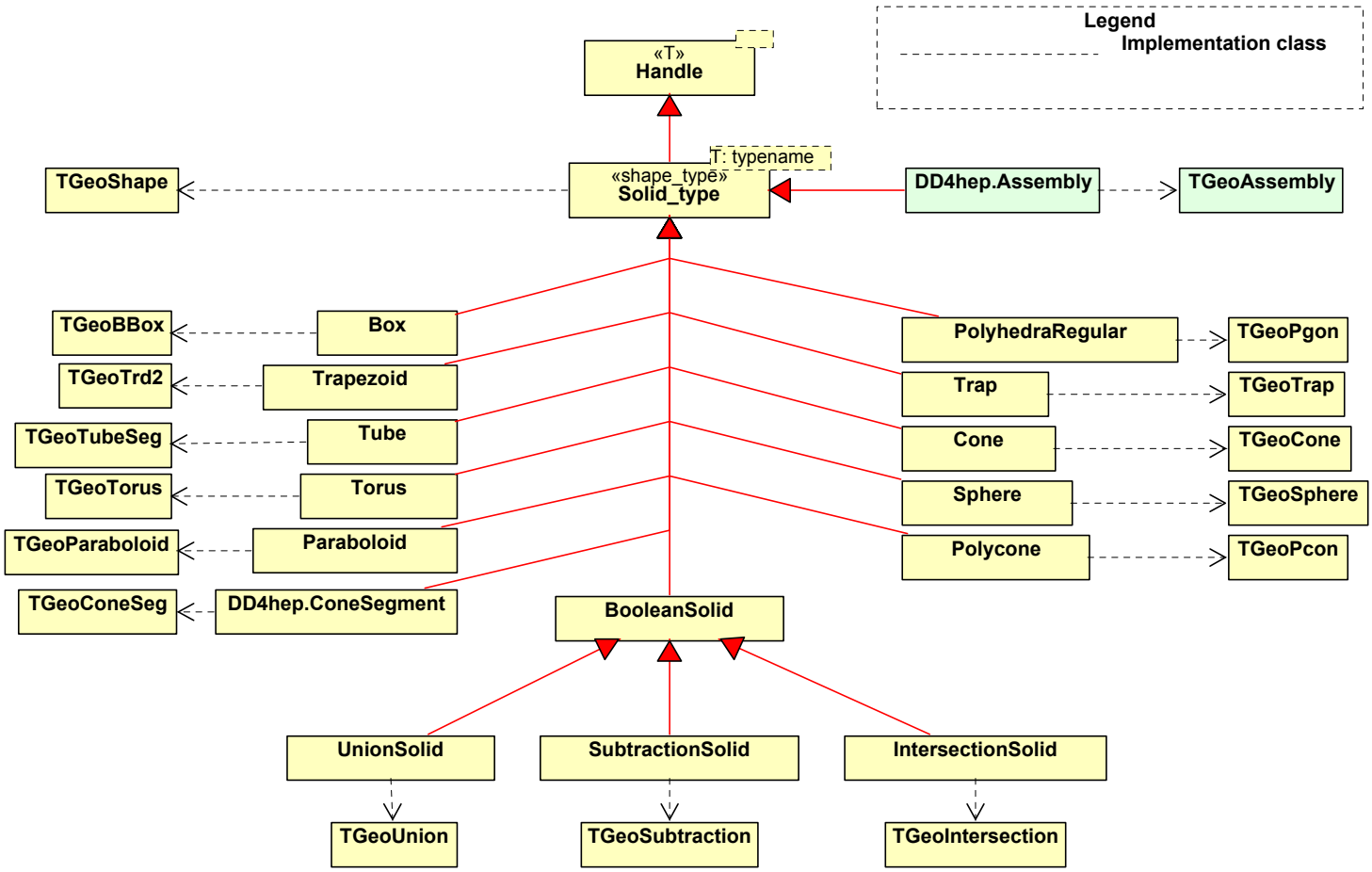
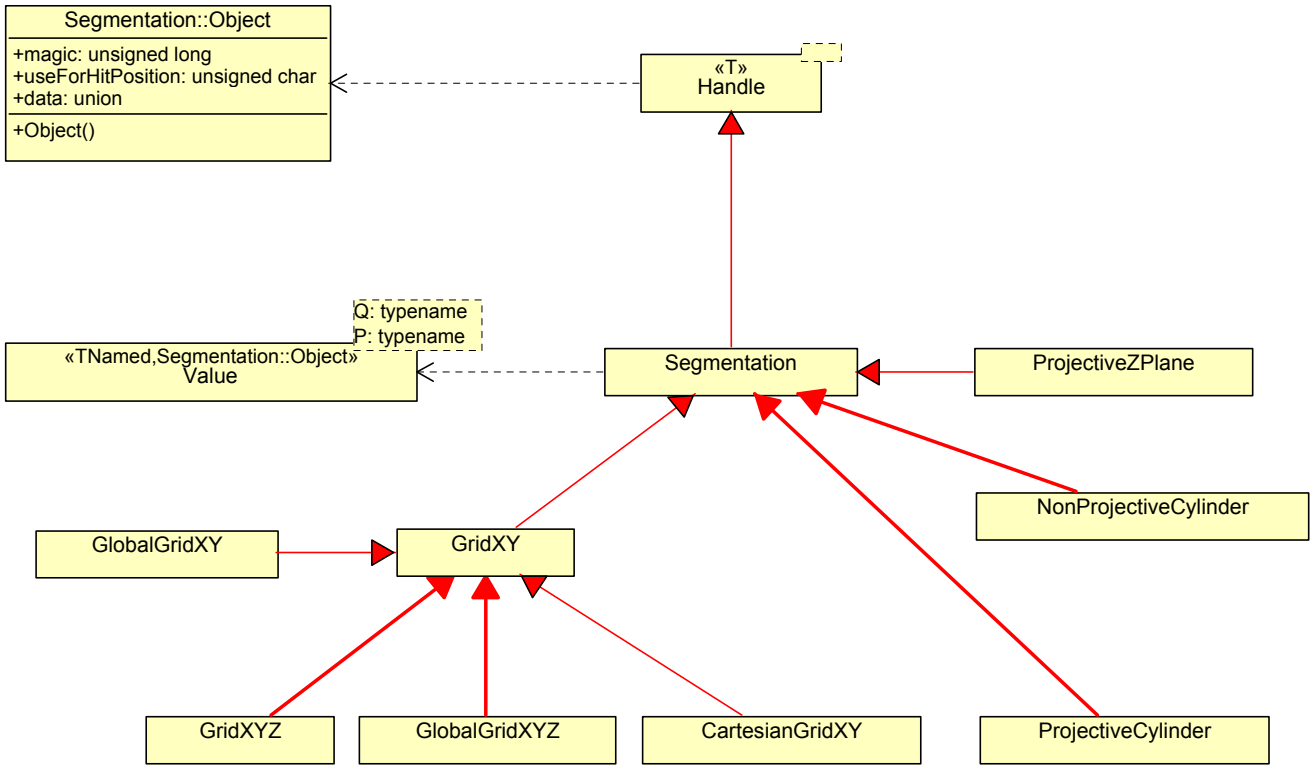
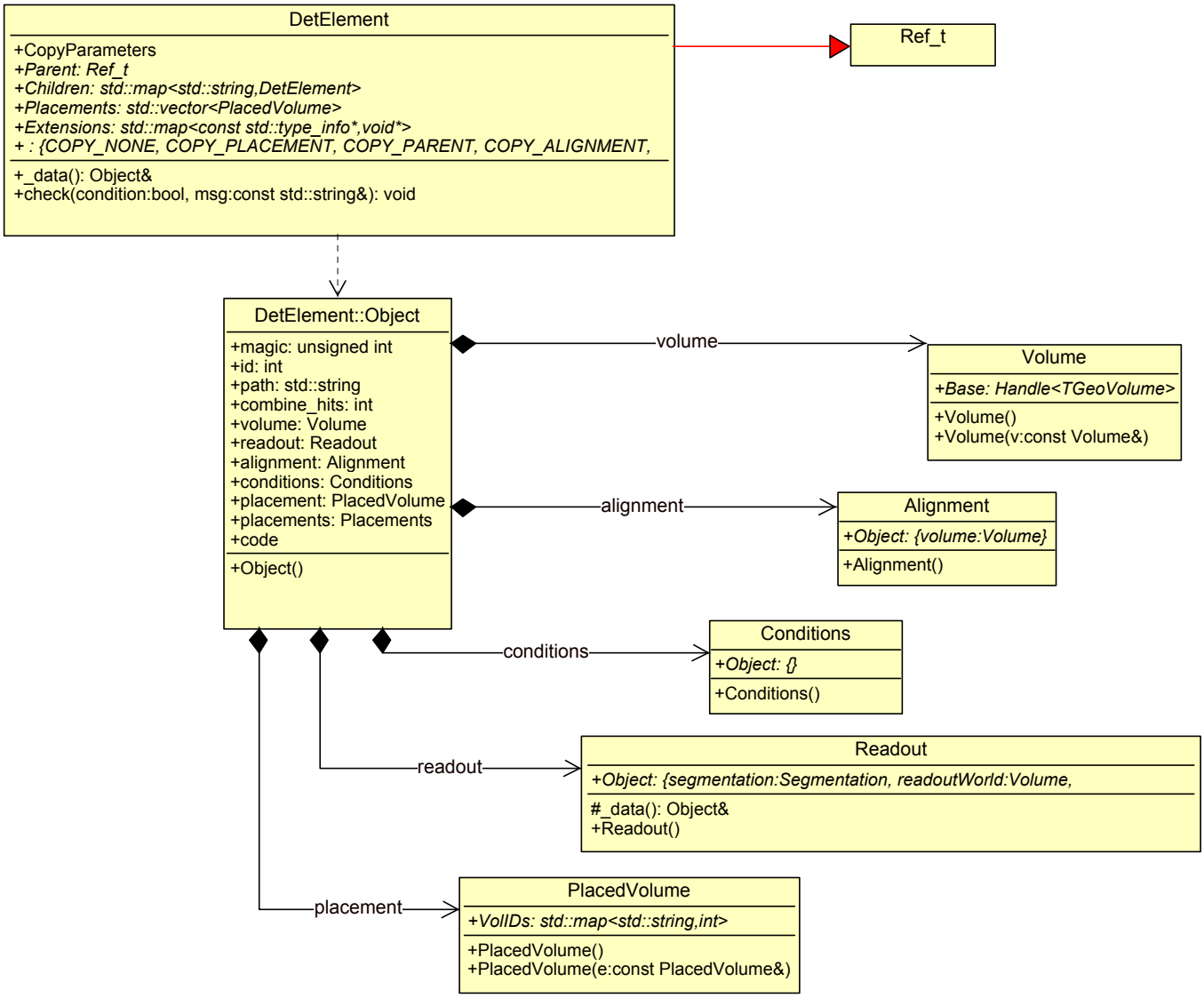

G4Region
production_cuts #0...n
G4ProductionCuts
region #0...1
G4FieldManager
field_mgr #0...1
user_limits #0...n
G4UserLimits
G4LogicalVolume
sens_det #0...1
G4VSDFilter
step_selector #1
G4VSensitiveDetector
hits #1...n
G4THitsCollection
readout #0...1
G4VReadoutGeometry

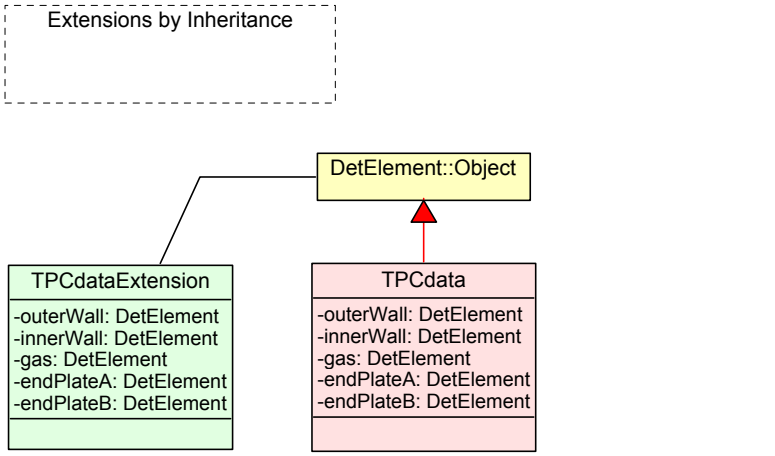

alignment #0..1
Alignment
conditions #0..1
Conditions
readout #0..1
Readout
visattrs #0..1
VisAttr
children #0..*
DetElement
Implemented by DD4hep
Implemented implicitly by ROOT
visattr #0...1
vis #0..1
VisAttr
placement #0..1
motherVol #1
PlacedVolume
Volume
limits #0..1
LimitSet
volume #1
region #0..1
Region
sens_det #0..1
SensitiveDetector
volID #0..*
solid #1
Solid
VolIDs
GDML content
material #1
Material

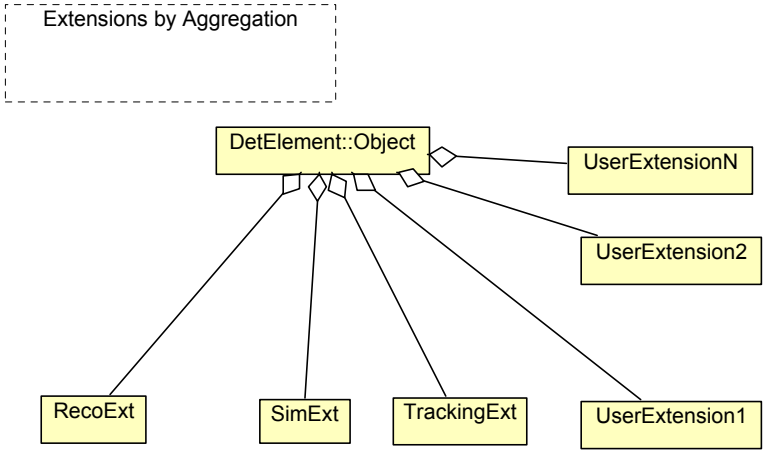

ExtensionEntry

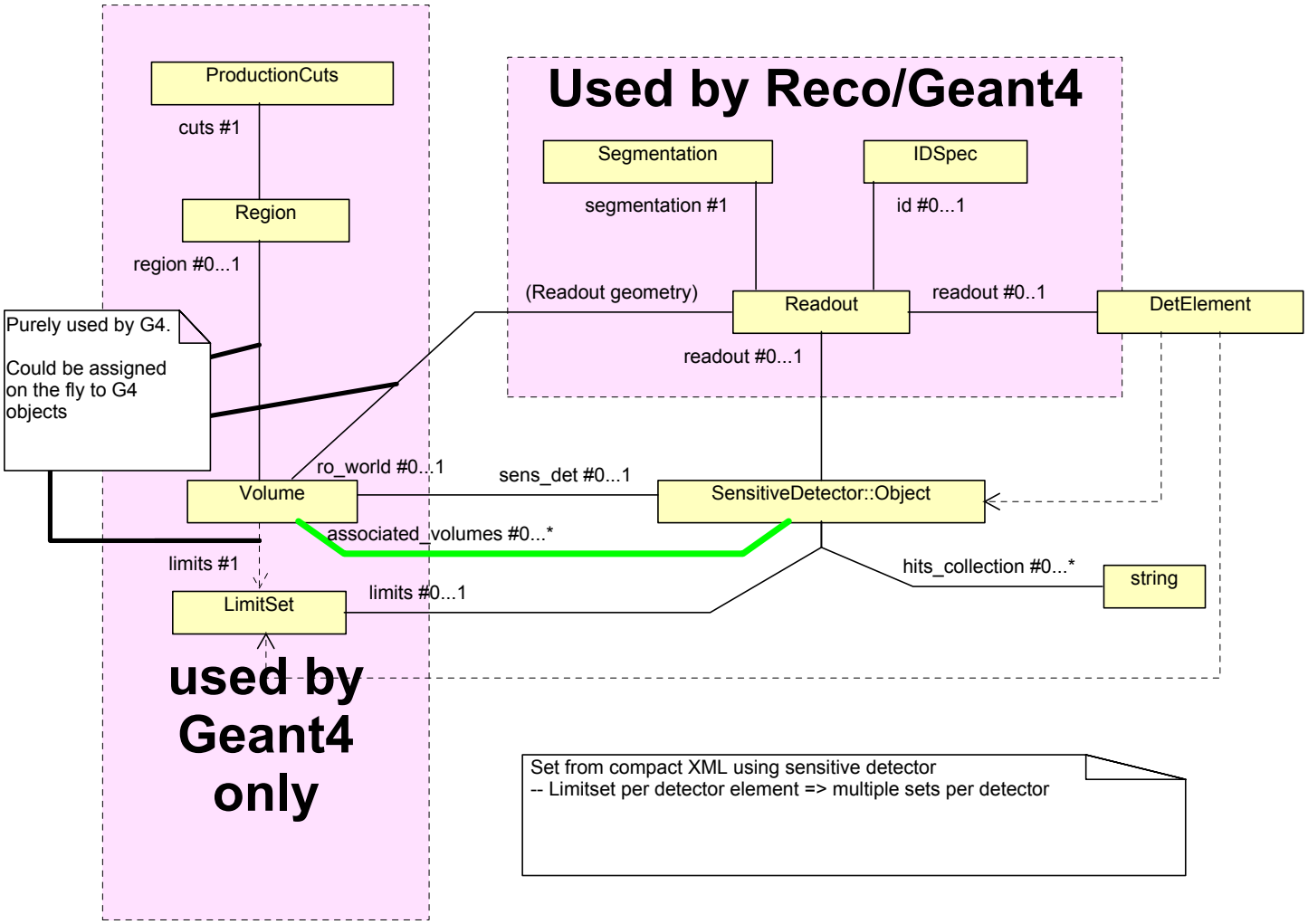

Legend
Implementation class
«T»
Handle
T: typename
«shape_type»
Solid_type
TGeoShape
DD4hep.Assembly
TGeoAssembly
TGeoBBox
Box
TGeoTrd2
Trapezoid
TGeoTubeSeg
Tube
TGeoTorus
Torus
TGeoParaboloid
Paraboloid
TGeoConeSeg
DD4hep.ConeSegment
PolyhedraRegular
TGeoPgon
Trap
TGeoTrap
Cone
TGeoCone
Sphere
TGeoSphere
Polycone
TGeoPcon
BooleanSolid
UnionSolid
SubtractionSolid
IntersectionSolid
TGeoUnion
TGeoSubtraction
TGeoIntersection

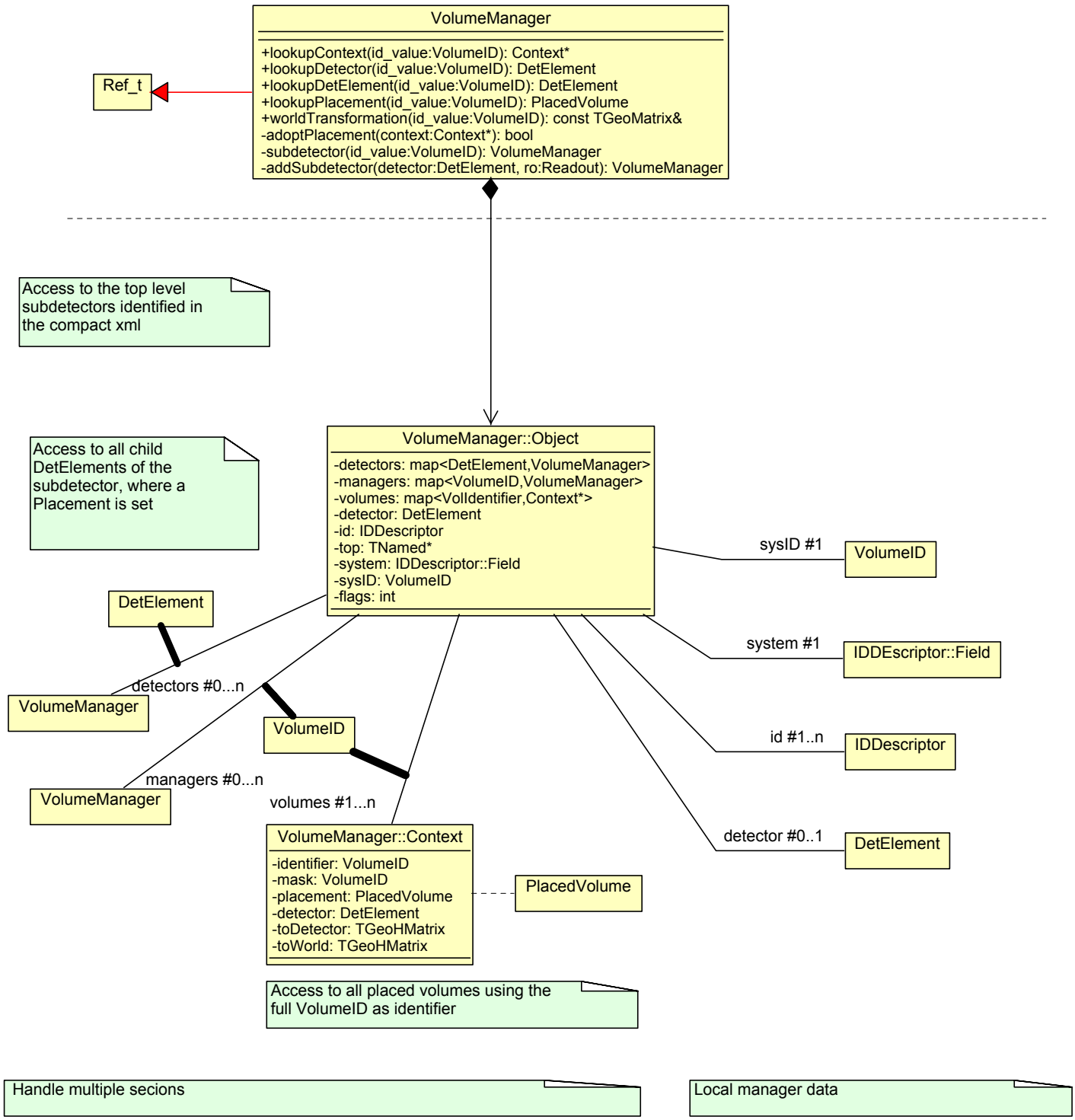

Segmentation::Object
+magic: unsigned long
+useForHitPosition: unsigned char
+data: union
+Object()
«T»
Handle
Q: typename
P: typename
«TNamed,Segmentation::Object»
Value
Segmentation
ProjectiveZPlane
NonProjectiveCylinder
GlobalGridXY
GridXY
GridXYZ
GlobalGridXYZ
CartesianGridXY
ProjectiveCylinder

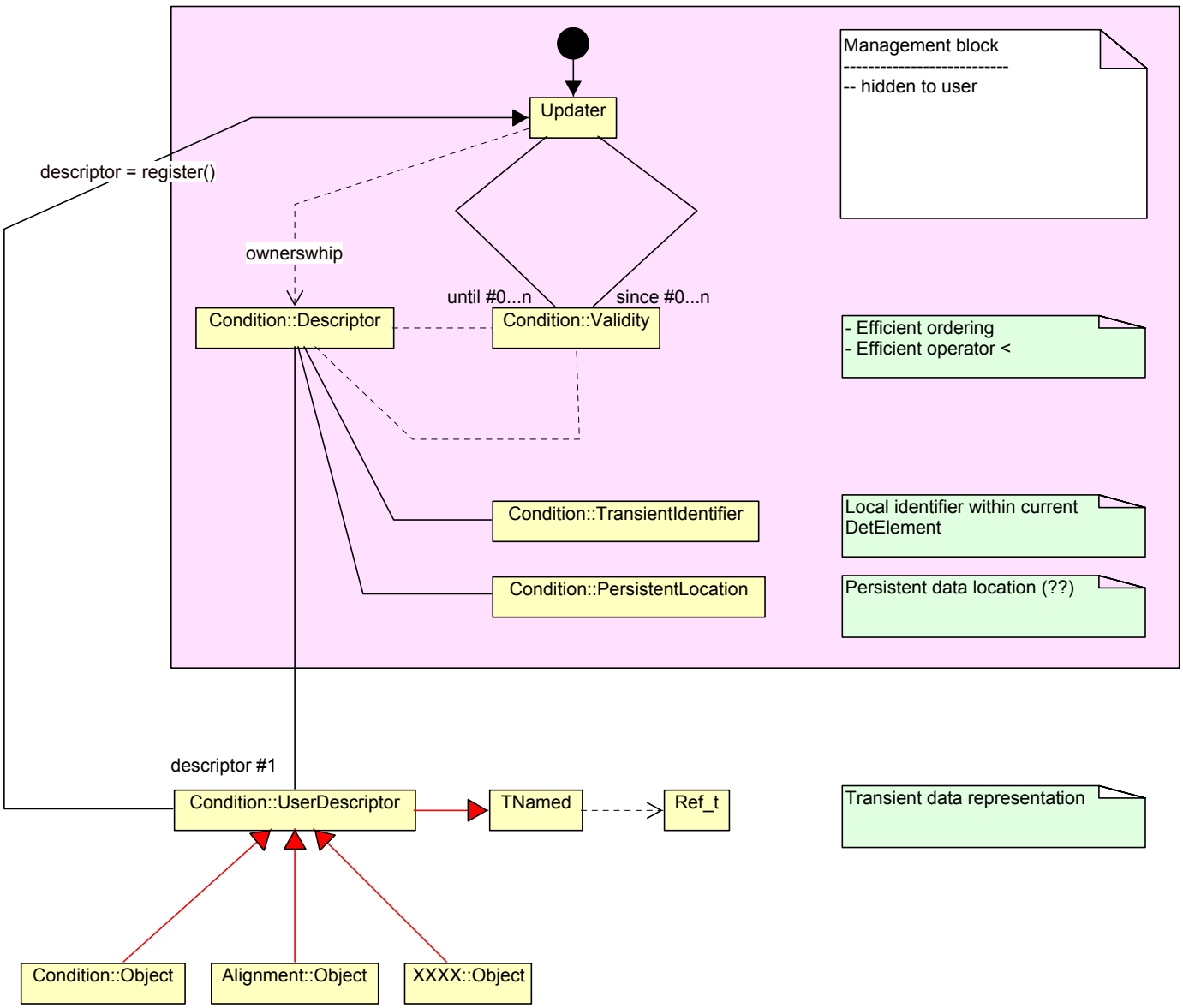

DetElement
+CopyParameters
+Parent: Ref_t
+Children: std::map<std::string,DetElement>
+Placements: std::vector<PlacedVolume>
+Extensions: std::map<const std::type_info*,void*>
+ : {COPY_NONE, COPY_PLACEMENT, COPY_PARENT, COPY_ALIGNMENT,
+_data(): Object&
+check(condition:bool, msg:const std::string&): void
Ref_t
DetElement::Object
+magic: unsigned int
+id: int
+path: std::string
+combine_hits: int
+volume: Volume
+readout: Readout
+alignment: Alignment
+conditions: Conditions
+placement: PlacedVolume
+placements: Placements
+code
+Object()
volume
Volume
+Base: Handle<TGeoVolume>
+Volume()
+Volume(v:const Volume&)
alignment
Alignment
+Object: {volume:Volume}
+Alignment()
conditions
Conditions
+Object: {}
+Conditions()
readout
Readout
+Object: {segmentation:Segmentation, readoutWorld:Volume,
#_data(): Object&
+Readout()
placement
PlacedVolume
+VolIDs: std::map<std::string,int>
+PlacedVolume()
+PlacedVolume(e:const PlacedVolume&)


Extensions by Inheritance
DetElement::Object
TPCdataExtension
-outerWall: DetElement
-innerWall: DetElement
-gas: DetElement
-endPlateA: DetElement
-endPlateB: DetElement
TPCdata
-outerWall: DetElement
-innerWall: DetElement
-gas: DetElement
-endPlateA: DetElement
-endPlateB: DetElement


Extensions by Aggregation
DetElement::Object
UserExtensionN
UserExtension2
RecoExt
SimExt
TrackingExt
UserExtension1

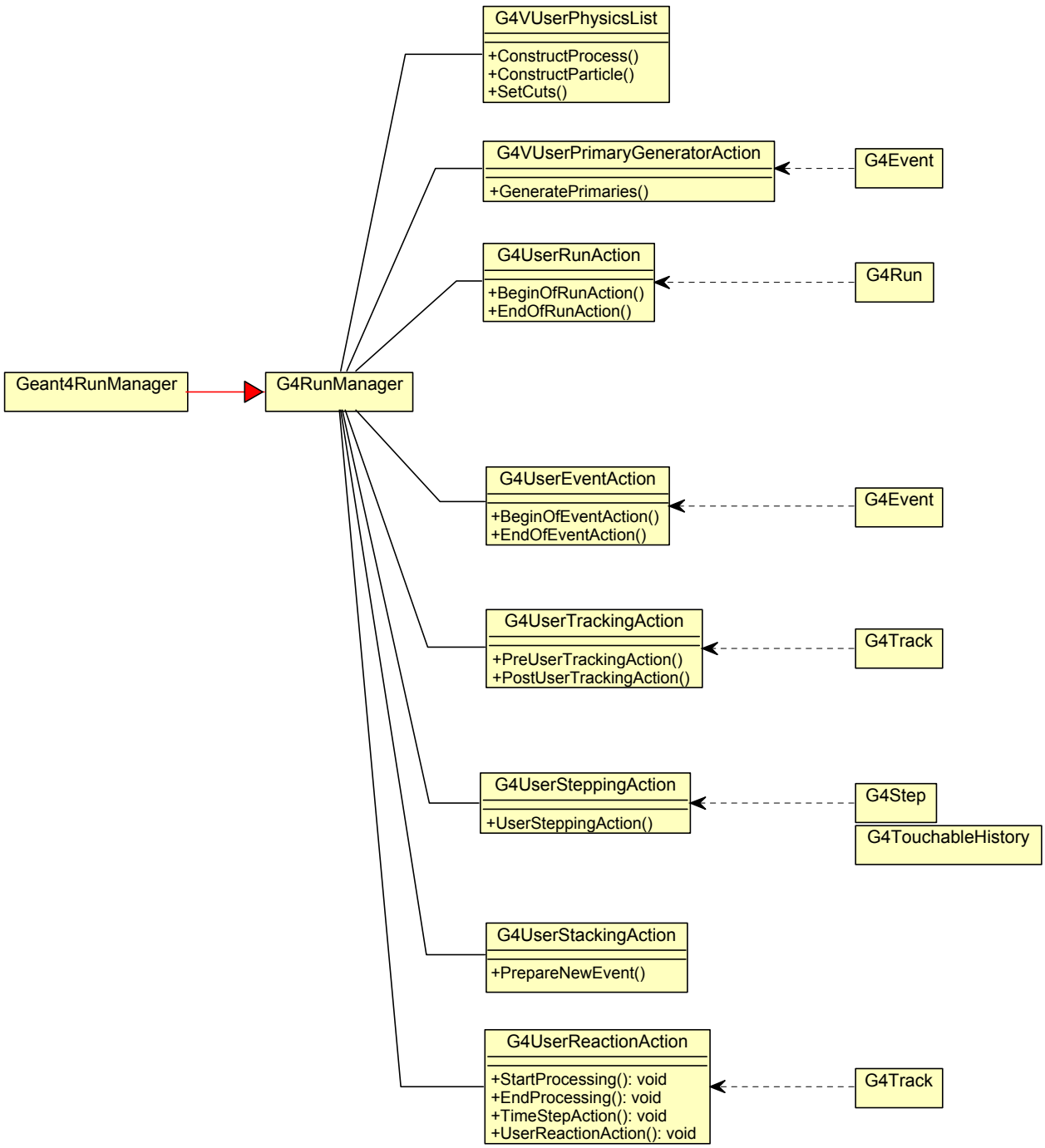

ProductionCuts
cuts #1
Region
region #0...1
Used by Reco/Geant4
Segmentation
segmentation #1
IDSpec
id #0...1
(Readout geometry)
Readout
readout #0..1
DetElement
readout #0...1
Purely used by G4.
Could be assigned on the fly to G4 objects
ro_world #0...1
Volume
sens_det #0...1
SensitiveDetector::Object
associated_volumes #0...*
limits #1
LimitSet
limits #0...1
hits_collection #0...*
string
used by Geant4 only
Set from compact XML using sensitive detector
-- Limitset per detector element => multiple sets per detector


VolumeManager
+lookupContext(id_value:VolumeID): Context*
+lookupDetector(id_value:VolumeID): DetElement
+lookupDetElement(id_value:VolumeID): DetElement
+lookupPlacement(id_value:VolumeID): PlacedVolume
+worldTransformation(id_value:VolumeID): const TGeoMatrix&
-adoptPlacement(context:Context*): bool
-subdetector(id_value:VolumeID): VolumeManager
-addSubdetector(detector:DetElement, ro:Readout): VolumeManager
Ref_t
Access to the top level subdetectors identified in the compact xml
Access to all child DetElements of the subdetector, where a Placement is set
VolumeManager::Object
-detectors: map<DetElement,VolumeManager>
-managers: map<VolumeID,VolumeManager>
-volumes: map<VolIdentifier,Context*>
-detector: DetElement
-id: IDDescriptor
-top: TNamed*
-system: IDDescriptor::Field
-sysID: VolumeID
-flags: int
sysID #1
VolumeID
system #1
IDDEscriptor::Field
id #1..n
IDDescriptor
detector #0..1
DetElement
DetElement
detectors #0...n
VolumeManager
VolumeID
managers #0...n
VolumeManager
volumes #1...n
VolumeManager::Context
-identifier: VolumeID
-mask: VolumeID
-placement: PlacedVolume
-detector: DetElement
-toDetector: TGeoHMatrix
-toWorld: TGeoHMatrix
PlacedVolume
Access to all placed volumes using the full VolumeID as identifier
Handle multiple secions
Local manager data


Management block
-- hidden to user
Updater
descriptor = register()
ownerswhip
until #0...n
since #0...n
Condition::Descriptor
Condition::Validity
- Efficient ordering
- Efficient operator <
Condition::TransientIdentifier
Local identifier within current DetElement
Condition::PersistentLocation
Persistent data location (??)
descriptor #1
Condition::UserDescriptor
TNamed
Ref_t
Transient data representation
Condition::Object
Alignment::Object
XXXX::Object

Geant4 provided user hooks

Any simulation program is supposed to overwrite the appropriate actions neccessary to perform the required functionality.
This means a flexible simulation program must hook into these callbacks and provide for each entry action queues to ease the implementation of small components.



Geant4Concept

General concept

Granularity:
-- One single G4 action entry leads to lengthy actions and spagetti-code.
Allow the seperation of each action into a sequence of individual action
each only serving a very special pupose and hence be very granular
To be seen: Is a mechansm necessary to interrupt an action sequence
and terminate the processing prematurely?

-- Sequencing:
Whereas for tracking-, stepping, run-, etc. actions the existing granularity looks sufficient,
for the event action sequences more "artificial" granularity may be desireable.

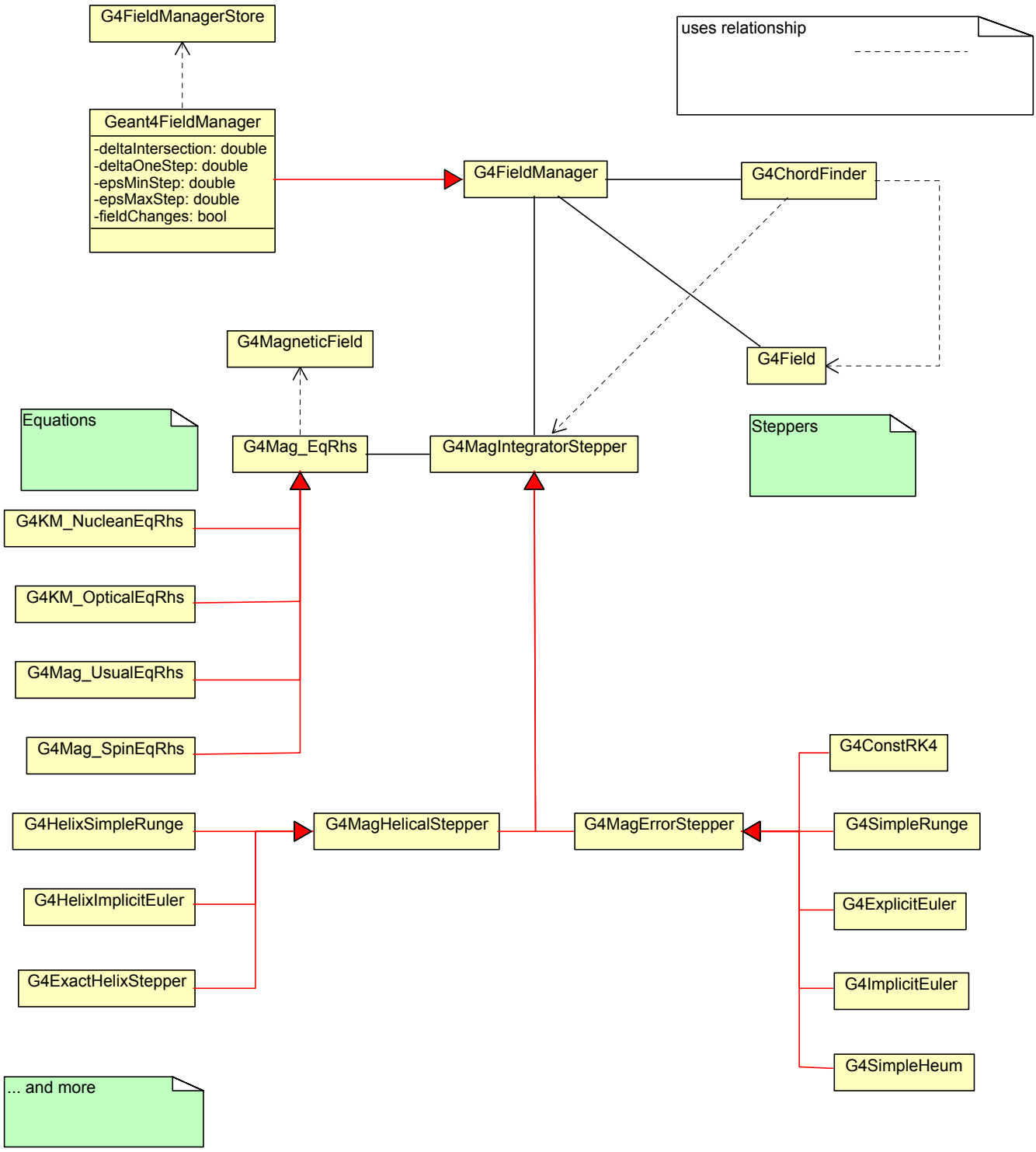
For example monitoring components of certain subdetectors
- wants to be initialized for each event,
- be called after processing the event
Such required functionality automatically leads to "Processing Phases",
where the simulation performed by Geant4 is only one of these.
Other phases may be the primary event generation or - as mentioned -
monitoring.

Important:
-- Any action may register for any phase supplying a member function as a callback
accepting the G4Event and the phase name as an argument

Since any action may register for any phase, this is sort of orthogonal to the action sequences,
which do not allow for calls of independent objects which do not belong to the same type of G4
callback.

If necessary the phase concept may be extended to runs.

-- Flexibility and Open-ness:
All actions must be simple.
Any setup functionality is outside the component.
Hence, various setup mechanisms may be attached: XML, python, Cint, ...


G4FieldManagerStore
uses relationship
Geant4FieldManager
-deltaIntersection: double
-deltaOneStep: double
-epsMinStep: double
-epsMaxStep: double
-fieldChanges: bool
G4FieldManager
G4ChordFinder
G4MagneticField
G4Field
Equations
G4Mag_EqRhs
G4MagIntegratorStepper
Steppers
G4KM_NucleanEqRhs
G4KM_OpticalEqRhs
G4Mag_UsualEqRhs
G4Mag_SpinEqRhs
G4ConstRK4
G4HelixSimpleRunge
G4MagHelicalStepper
G4MagErrorStepper
G4SimpleRunge
G4HelixImplicitEuler
G4ExplicitEuler
G4ExactHelixStepper
G4ImplicitEuler
G4SimpleHeum
... and more