The GEANT4 Experience — Panel Discussion on Transition to OO — CHEP97

John Allison
University of Manchester, UK

9th April 1997

The GEANT4 Visualization
John Allison, University of Manchester
Presented at CHEP97, Berlin
9th April 1997

1. The Early Days
2. The G4 Components
3. CVS
4. GNU Make (gmake)
5. CLHEP
6. Container Classes
7. Design Patterns
8. The Early Releases
9. The Alpha Release
10. Thoughts on the Product
11. The Down Side
12. Thoughts about People
13. Summary

1. The Early Days

- OOP was new to most of us
- CVS was new to most of us
- (g)make experience was limited

2. The G4 Components

Snapshot last week. (Some re-grouping of categories has taken place since then.)

Category	Files		Lines	
	.hh	.cc	.hh	.cc
digits+hits	18	11	863	967
events+tracks	22	17	1054	1407
g3tog4	10	41	400	2969
geometry	67	177	6127	91414
global	69	57	4810	4999
graphics_reps	23	19	1655	2510
intercoms	8	7	331	610
interfaces	3	6	104	1349
particle+matter	168	147	12904	35670
tracking	14	10	1454	1785
visualization	59	67	3649	15253
.icc files	79			
	616	632	37057	172154
	1248		209211	

Note that visualization does *not* include DAWN and OPACS stand-alone packages, nor the Hepvis code.

Note also that no. of lines is not a criterion for productivity!!

3. CVS

- Indispensable
- Used also as a development tool
- Thwarted somewhat by
  - lack of bandwidth
  - lack of courage
  - CERN policy (rsh access)

  but still an indispensable development tool
- Under-used — no branches
- Mis-used — .ps files
- Major reorganisations need good planning

4. GNU Make (gmake)

- Each component and sub-component has a GNUmakefile
- Each includes GNUmakefile.architecture
- Each includes GNUmakefile.common
- "Higher" makefiles: cd subdir; $(MAKE)
- Inadequacies now showing
- Will probably be re-engineered along BaBar lines, including BaBar's release structure

5. CLHEP

- HEPGeometry:

G4LorentzRotation.hh	G4LorentzRotation.icc
G4LorentzVector.hh	G4LorentzVector.icc
G4Normal3D.hh	G4Plane3D.hh
G4Point3D.hh	G4Transform3D.hh
G4Vector3D.hh	Hep3Vector.hh
Hep3Vector.icc	HepRotation.hh
HepRotation.icc	

- HEPRandom:

DRand48Engine.hh	HepJamesRandom.hh
HepRandom.hh	HepRandom.icc
HepRandomEngine.hh	HepRandomEngine.icc
RandBreitWigner.hh	RandEngine.hh
RandExponential.hh	RandFlat.hh
RandFlat.icc	RandGauss.hh
RandPoisson.hh	Randomize.hh
RanecuEngine.hh	RanluxEngine.hh
SeedTable.hh	

- *Extremely useful*

- GEANT4 team have made a big contribution

6. Container Classes

I.e., lists, maps, etc.

- Rogue Wave tools.h++
- (Will probably migrate to STL)
- E.g., RWTPtrOrderedVector<G4VView> (view list)
- E.g., RWTValOrderedVector<G4Point3D> (polyline)
- *Enormously* helpful
- Takes 90% of the worry out of storing objects

7. Design Patterns

- Limited use
- Useful for communicating function of class
- Obscure (personal opinion)

8. The Early Releases

- Much interest shown
- Ported to Windows NT
- BaBar "took it on board"
- Not at all a usable product
- Much re-design, reorganisation

9. The Alpha Release

April 1997

- Even more interest shown
- Public, but...
- Not for use!
- No guarantees
- No support
- At your risk!
- For GEANT4's benefit
- Friendly users identified
- Friendly users helped
- Feedback expected
- Feedback needed

Help us get this show on the road.

The beta release is planned for end 1997/ early 1998.

10. Thoughts on the Product

- GEANT4 is highly object oriented
- GEANT4 contains some highly sophisticated OO designs, particularly in geometry, tracking and intercoms
- GEANT4 contains a lot of physics, encapsulated in particle and process classes
- The abstract visualization interface seems to work
- The OO Method is essential
- Categories developed more or less independently work together without serious conflicts/clashes/adverse interactions
- Abstract interfaces work
- Inheritance and code re-use works
- OOP leads to high number of small code units (functions)
- OOP leads to large number of files
- CVS essential
- OO Design tools have been used successfully
- Performance is good

11. The Down Side

- Code is large — 8 MB on Linux, 40 MB on AIX (with debug tables)!
- Compilation takes 1 – 2 hours!
- OOP leads to high number of small code units (functions)
- OOP leads to large number of files
- Design tools needed
- More sophisticated compilers/linkers needed

12. Thoughts about People...

- ...thoughts about the GEANT4 Team

  In spite of initial inexperience on the part of most members (only one or two had previously been taught OO)...

  - All were experienced and established in computing
  - All were/are committed to OO
  - All were willing to put effort into learning OO
  - Most learnt "on the job"
  - All are now adequately proficient in C++ OO

- ...thoughts about physicists in general
  - Physicists *can* learn OO Methods
  - There has to be a way of *easing* people into it.
  - Physicists are good at learning on the job
  - Needs dedication to learn full OO
  - Can use "recipe" methods to begin with
  - Start with "user defined types"
  - Needs 2 years to grasp concepts of OO and adequate mastery of language!
  - Need to acquire new way of thinking
  - It just comes with practice — difficult to teach
  - It comes more naturally to some than others
  - Once acquired, one feels it is the "natural" way of thinking
  - Young people will know nothing else
  - Fun

13. Summary

It's definitely the way to go!