GEANT4:
Status and Plans

H.P. Wellisch   CERN - CH

ATLAS Software week

Sep 04 1997

Introduction

- **The collaboration:**
  - >10 experiments involved
  - ~40 labs/institutes/universities
  - ~100 engineers, physicists, and computer scientists
- **Objective:**
  - *OO toolkit for detector simulation in HEP*
- **Time-frame: from end 1994 to end 1998**

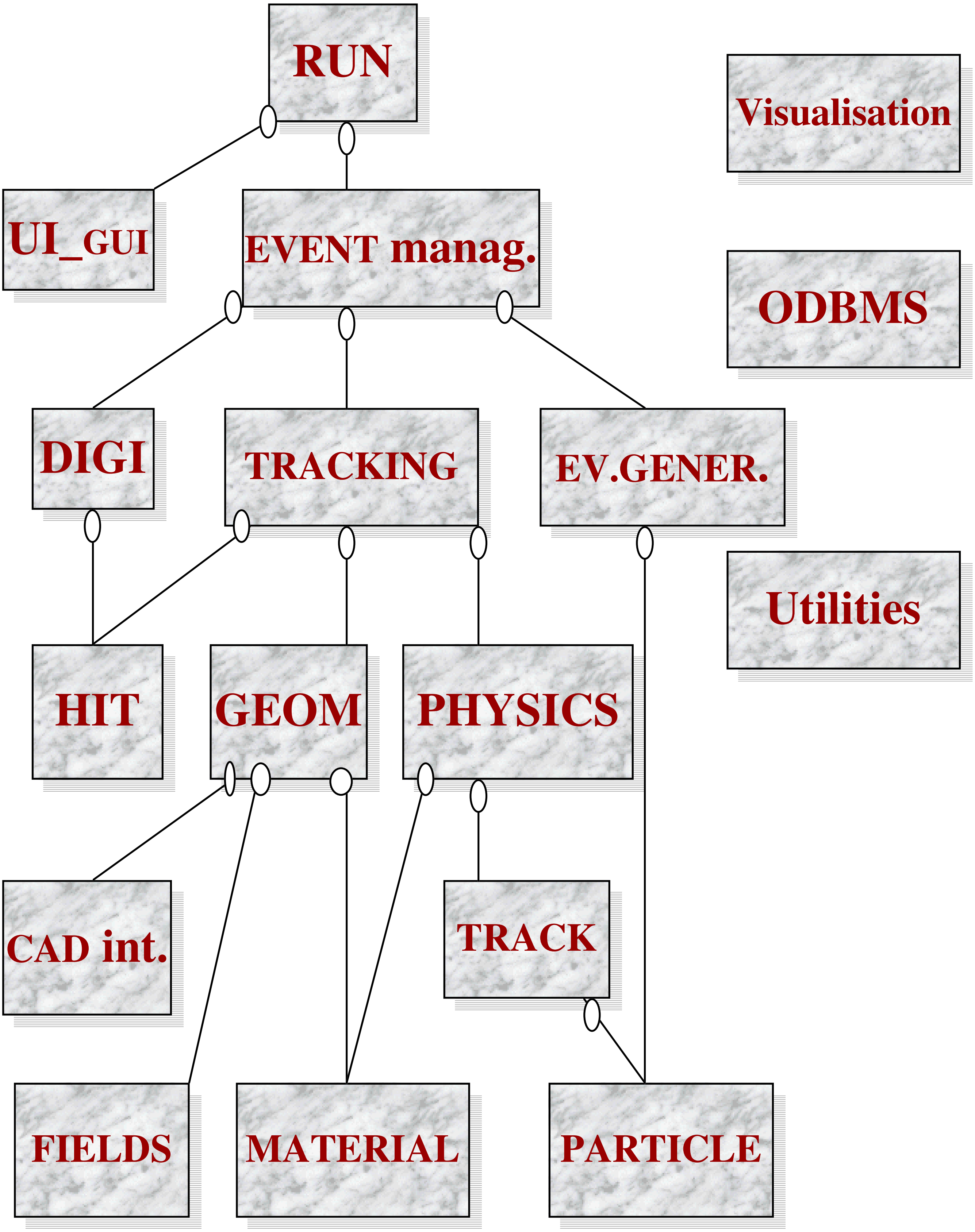
Some History

- **Milestones at end '95:**
  - OO methods, problem domain, full OOA&D
  - tracking prototype, performance
- **Milestones at spring '97**
  - alpha version comparable to GEANT 3.21
  - ODBMS persistency (hits)
  - transparency of physics models for users and developers

Our Strategy

- **Software Engineering in HEP**
- **The software process: *focus on distributed development***
- **ESA PSS-05: URD v.06**
- **Object-Oriented methodology: Booch+unified**
- **OOA&D: Rational Rose**
- **QA: Insure, Logiscope, code inspection, coding guidelines, scripts**
- **Testing at class-level (ex: 375 test-cases for low E hadronics)**

Class Category Diagram



RUN

Visualisation

UI_GUI

EVENT manag.

ODBMS

DIGI

TRACKING

EV.GENER.

Utilities

HIT

GEOM

PHYSICS

CAD int.

TRACK

FIELDS

MATERIAL

PARTICLE

Geometry and Field

- **ISO STEP compliant solid modeler (CAD systems)**
  - Volumes independent from solids representations: CSG, BREP, SWEPT
  - Parameterised volumes (dimensions+transformation)
- **Navigator: performance in CAD geometries**
- **Converter for GEANT3 geometries**
- **OO design for integration of motion in any fields**
  - Several 5-th order RungeKutta methods
- **Plans'98: *Booleans, parameterised volumes by solid or material, multiple turns in one step, exchange with CAD***

Runs, Events, Tracks

- **Full functionality for Runs+Events+Tracks**
  - Multiple ‘physics runs’ in one job with changing geometries, materials
  - Tracks stacking mechanism: 3 priority levels (From trigger studies to slow particles)
- **Event generators interface:**
  - ASCII Hepevt + PYTHIA
- **Plans ’98: *OO/C++ generators: Fritiof++, Lund++***

Tracking

- **Full functionality for tracks and stepping**
- **OO design kept hierarchical:**
  - aggregation and inheritance
  - design for both detailed and fast simulation
- **Generalised interactions actions:**
  - AlongStep (distributed in space-time)
  - PostStep (interaction at space-point)
  - AtRest (interaction at a time, stopping particles)
- **Plans End ‘97: *fast shower parameterisations***

Physics OO Design

- **Limit step:**
  GetPhysicalInteractionLength()
- **Compute final state: DoIt()**
- **Independence from physics units**
- **Transparent access to cross-sections from files, tables, analytical formulae**
- **Multiple implementation of physics models, for processes and channels, by energy range, materials/elements and particle types (had. physics)**

Hits/Digitisation

- **Hits (user defined) save data as**
  - space-time, E deposition, particle info, detector ID, etc..
- **Hits collections represent the logical read-out**
- **Sensitive detectors can be made by volumes, solids, and ‘touchables’**
- **Read-out, parameterisation, reconstruction geometries can be independent of tracking geometry**
- **Plans ’98: *Hits/Digitisation examples***

Object Persistency

- **ODBMS solution via RD45 (Objectivity)**
- **Tracker-type hits vs. calorimeter-type hits**
  - Nobjects = Naccess vs. Naccess >> Nobjects
  - Store each hit as a separate persistent object
  - Store all hits of each type as persistent array
- **Minimal performance and storage overhead**
- **Minimal modifications of implementation**
- **Plans'98: *Trajectories*, *Runs, Events, Geometry***

GUI and Visualisation

- **Visualisation and GUI OO abstract interfaces**
- **Visualisation drivers and GUI prototypes:**
  - OGL, Inventor, X11, PostScript, DAWN, OPACS, VRML (see http://arkoop2.kek.jp/GEANT4/vrml for some nice pictures)
  - **Command-line,Wo, Tcl/Tk, Tcl/Java, batch+macros**
- **Drawing functionality available for:**
  - **detector, trajectories, steps, hits**
- **Plans'98:**
  - **Visualisation:** ***selections, cut-views, ray-tracing***
  - **GUI*: picking, XVT, JavaPVM***

User Actions

- **9 user actions in total**
- **Mandatory actions (abstract classes):**
  - event generation
  - detector construction
- **Optional actions (default behaviour):**
  - run, event, track, step, stacking actions
  - processes-lists, particles-lists
- **Main: initialisation and**

“Beam-on”

Current Functionality

- ISO STEP geometry + fields + RK integrators
- Runs, events + HEPEVT, pile-up, tracking
- EM showers as G3; low energy; optical photons
- Hadronic showers vs G3; models by energy range, material, element, particle
- Calorimeter + tracker hits; ODBMS persistency
- Visualisation design: DAWN, OGL, OPACS, OI, VRML, PS, X11
- 9 classes for user intervention + main()
- ~300000 lines of code in C++; ~600 C++ classes

To put it into a nut-shell

- **Alpha-02 comparable to GEANT3**
  - *Physics, Tracking, and Geometry*
- **Hits are persistent using ODBMS technology**
- **OO design enables us to provide transparency of physics**
- **Alpha-01 checked by alpha-testers: Alpha-02; checking continues**
  - *ATLAS, CMS, ALICE, BaBar*

Plans `98

- **Produce *Beta-version***
- **Object-persistency for events and geometry**
- **Visualisation, GUI, *beta*-documentation**
- **Demonstrate fast MonteCarlo capabilities**
- **Use CAD geometries, extend topology (STEP)**
- **Extend validity range of physics models**
    - **high energy: generator models, muons...**
    - **low energy: neutron transport, transition radiation...**

Conclusions

- GEANT4 *Alpha-02* released in Summer ’97
- Software Engineering tools, workshops, and good design play a fundamental role in the software process
- OOP is a key factor for delivering simulation software that users can understand + enhance
- Transparency of physics design + implementation improves the verification and reliability of results
- ODBMS technology provides a performant and reliable object-persistency service
- RD44 can produce a GEANT4 *Beta-version* in ’98