

# History of Geant4

5<sup>th</sup> Geant4 Space Users' Workshop

*@Univ. of Tokyo*

*2008.02.12*

**Katsuya Amako**

(KEK)

# *Aim of the Current Talk*

- By giving you a brief history of the Geant4 development , I'll tell you the design decisions we made during the R&D stage of the Geant4 simulation toolkit.
- This explains you why the toolkit has its current shape.
- And knowing this may eventually help you to find a better way to use it for your applications.

# HEP in Early of 1990

## ■ Why talk about High Energy Physics (HEP)?

- In early days the Geant4 development was driven by HEP physicists and engineers.
- The development of Geant4 is in parallel with the history of HEP.

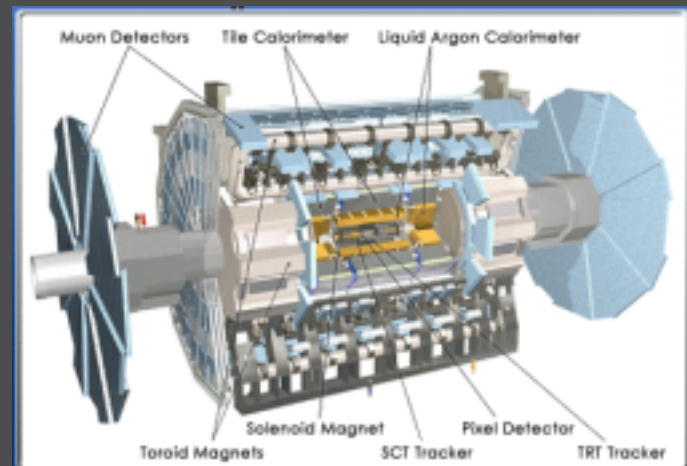
## ■ The LHC project at CERN

- Early of 1990 was the period when the preparation of LHC was kicked off.
- Example: Letter of Intent of ATLAS experiment was submitted in 1992.



### LHC:

- proton+proton collider
- Energy: 7Tev + 7Tev
- Luminosity:  $1 \times 10^{34} \text{ cm}^{-2}\text{s}^{-1}$



### Atlas Experiment: (numbers in 2007)

- # of scientists: ~1800
- # of institutes: ~ 164
- # of countries: 35

# Software Engineering

---

- Because of huge scale of the LHC experiments we foresaw the following serious difficulty in managing our software systems:
  - Scale and complexity
  - Long life cycle (more than 20 years)
  - Large scale of manpower necessary for development
  - Geographically wide spread people
  - Fluidity of people in the course of development and maintenance
  - For example....
  
- **Software development and methodology**
  - For the development of a complex LHC software system, we considered it was absolutely essential to employ an engineering discipline.
  - We surveyed for applicable software engineering methodologies at that time (more than 15 years ago!).
    - ◆ It seemed that the 'Object-oriented Design' approach was most promising.

# Necessity of R&D for OO Software Engineering

---

- Without seeing a clear advantage people won't move to a new technology
  - HEP didn't have any experience in using OO methodology at that time. Only we knew was the non-disciplinary approach based on Fortran.
  - To convince LHC and also HEP researchers to move to this new technology, we needed to provide a proof of merit of utilizing the OO approach.
  - A serious R&D was necessary.
    - A demonstration of OO application to a small scale program wouldn't convince people. We needed a large scale HEP software.

→ *GEANT*

# GEANT3 – Predecessor of Geant4

---

## ■ History of GEANT3

- GEANT and GEANT2
  - ◆ A bare FORTAN based framework for tracking particles.
  - ◆ 1<sup>st</sup> version created in 1974 by R.Brun et al. at CERN.
- GEANT3 - The final major version of the old GEANT series
  - ◆ Created in 1982.
  - ◆ Last version is GEANT3.21 released in 1994.
  - ◆ Maintenance frozen after the Geant4 public release.

## ■ GEANT3 Features

- A pure CERN product of about 200,000 lines of code created at least 50 man-years of work, spread over more than 15 years.
- De-facto standard for HEP detector simulation framework before Geant4.
- Not flexible enough to provide functionality for diverse requirements from HEP (LHC, heavy ions physics, neutrino physics, etc) and also other research domains (medical and biological science, astronautics, radio-protection, etc).
  - ◆ For example, ~60 routines need to be modified to add a new geometrical shape.

# R&D Projects for LHC Software

---

## ■ Reengineering of GEANT3 by OO approach

- Once we succeeded in developing an OO based detector simulation package that outperforms GEANT3, then we could convince the LHC community that OO is the way to pursue.

## ■ Two R&D projects around 1993

- Object-Oriented Analysis and Design of GEANT at KEK (ProdiG project)
- Investigation of class hierarchy for GEANT at CERN

## ■ R&D Proposal toward Geant4

- The members of above two projects agreed to merge their activities and to submit an R&D proposal to DRDC/CERN in August 1994.
- The number of proposal authors was *29 from 19 institutes and 9 countries*.

→ *International Collaboration*

(cf. GEANT3 - pure CERN product)

## ■ Thus Geant4 was kicked off in August 1994.

## ■ R&D Project name: RD44

# The Geant4 R&D Proposal

---

- The motivation and purpose of the project (from the proposal)
  - For the LHC and heavy ion experiments, an even larger degree of functionality and flexibility is required in GEANT, thus making necessary a re-design of the program.
  - Investigate the use of object-oriented techniques to enable us to meet these goals.
    - *This is the reason you see current Geant4 in OO.*
  - The philosophy (of the design) should not be 'GEANT simulates all the known physical processes at any energies', but it should rather be 'the simulation of any physical process at any energy can be easily plugged into GEANT *by any user*'.
    - *Geant4 is a toolkit and not an all-in-one and ready-to-use system.*
    - *You can plug in your own physics process.*
    - *You need to provide a 'PhysicsList' by yourself.*
  - GEANT is increasingly requested for application such as tomography, dosimetry, space science, etc. and the enhancements could also be used to advantage by people working in those fields.
    - *Geant4 tried to integrate requirement not only from HEP but also other research domains from the beginning of the project.*

# Disciplinary Design of Geant4 – OO Analysis and Design

---

## ■ Methodology employed

- After surveying wide variety of methodologies available in early 1990 we decided to adapt (not adopt) both J.Rumbaugh's "OMT" and G.Booch's so-called "Booch" method.
  - The base of this decision was that these approaches seemed to be pragmatic and usable even for non-expert like us.
  - These two methods evolved since then and it is now UML (Unified Modeling Language) which is an ISO standard.

## ■ Disciplines

- Don't rush for coding but do a design.
- User requirements document
  - ◆ Collect user requirements (*ESA PSS-05*)
- OO Analysis for global design
  - ◆ Find major objects necessary for the toolkit.
- OO Design for more detail design
  - ◆ Define relations and interactions of these objects. Also find more low level objects and their relations.

### [Note]

In the real world of design, we found that OOA and OOD were concurrent activities.

# Requirements Document and 3 Types of Users

---

## ■ Three types of users from the G4 developers' view point

### 1. Domain specific application users

Ex) Atlas detector simulation users

User Document: *Provided by the simulation developers*

### 2. Domain specific application developers (=G4 users)

Ex) Atlas detector simulation developers

User Document: *Users' Guide: For Application Developers*

### 3. Functionality enhancement developers (=G4 user)

Ex) Add a domain specific physics process

User Document: *Users' Guide: For Toolkit Developers*

## ■ User Requirement Document

- Taken into accounts these three types of user

- In ESA PSS-05 format

- <http://cern.ch/geant4/OOAandD/URD.pdf>

# OO Analysis/Design for Geant4

## ■ The path we took:

1. For identifying major objects/classes in Geant4, *we analyzed the GEANT3 user document.*

(Ex) event, vertex, particle, trajectory, volume, track, volume, sensitive-detector, material, interaction, box, tube,.....

2. After picking up major objects, we defined their relations and how they interact each other.

- Object diagrams (object relations)
- Scenario diagrams (object Interactions)

*Here we reused the algorithmic know-hows accumulated in GEANT3.*



Ex) How to categorize particle interactions,

How to organize geometrical shapes, etc

3. To cluster classes which have a closely coupled relation

- *Class category diagrams*

# Category Diagram

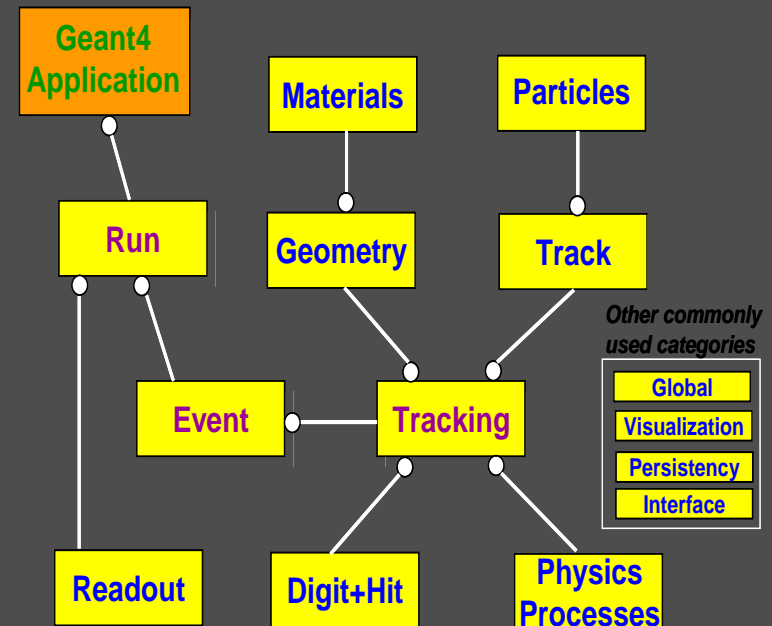
- 'Class Category Diagram' had a fundamental importance in the Geant4 collaboration.
  - Objects in a class category have strongly/cohesively coupled
  - One class category to other is weakly/loosely coupled
- This cohesive and loosely coupled nature enabled us to implement each category relatively independent.
  - We used each class category as a unit to share tasks in OOD (and also in the implementation phase)

→ *Essential for worldwide collaboration*

- Geant4 codes you can download from our distribution site are organized under the scheme of this category diagram.

## Category Diagram (Ver. 1998)

Note: Most categories defined in the 1<sup>st</sup> year



# Fundamental Decision in the Design Stage - 1

---

## ■ Most fundamental and challenging design in Geant4

- How to design 'particle' and 'interaction' in OO way?

## ■ OO purist vs. pragmatist approach

- **Purist:**

Particle is obviously an object in the real/natural world. Why not to mimic the nature?

- Move by itself, interact by itself, decay by itself,.....
- Active object!

- **Pragmatist:**

Simulation is not the real world

- User wants to access to simulation information on fly.
- The user wants to control artificially the particle fate, etc
- Passive object!

→ *Which design should we choose?*

# Fundamental Decision in OO Design Stage - 2

---

## ■ Decision

- A substantial time of the basic design phase was eaten by a long and serious debate of this choice.
- The final decision was made by following our basic guidance – sticking to pragmatic approach.
- Final decision: *Take the pragmatist's approach*

## ■ This decision introduced the design pattern you often find in Geant4: Passive objects controlled by an active object manager

- Step ↔ SteppingManager
- Track ↔ TrackingManager
- Event ↔ EventManager
- Run ↔ RuntManager
- Process ↔ ProcessManager
- Visualisation ↔ VisualisationManager
- .....

# Prototype coding – Selection of Language

---

## ■ Language independence in OOA/OOD

- OOA and OOD could proceed without specifying its implementation language.
  - ➔ Take note that an OO design could be implemented either in OO or procedural/non-OO languages.
- Anyway for the prototype coding we needed to select it.

## ■ A language war

- Historically many struggles/fights in the HEP community
  - ◆ In 70s: Fortran vs. Mortran
  - ◆ In 80s: Fortran vs. Pascal/Ada/Modula
- More complicated in early 90s – various HEP R&Ds to new languages
  - ◆ Fortran90
  - ◆ C
  - ◆ C++
  - ◆ Eiffel
  - ◆ Objective C
- *Decision: Follow the industrial trend*
  - ➔ C++
  - ◆ A twilight of HEP in computing!

# Prototype coding – Reuse and Standard

---

## ■ Reuse of OO codes from other R&D projects

- CLHEP codes
  - ➔ *This introduced the dependency to CLHEP*
- Object persistency codes by RD45 (the sibling R&D of Geant4)
  - ➔ *This introduced the dependency to Objectivity though it became obsolete.*

## ■ Stick to standard tools

- CVS
- GNUmake
- STL (in the early R&D phase, we used Rogue Wave Tools.h++)
- X11
- OpenGL
- PostScript
- VRML
- CAD Interface – Step standard (*currently not available*)

# Performance of Prototype code

---

## ■ Prototype code

- Based on the OO design, the kernel part (particle transportation and related controls) was implemented by C++ in the 1st year of R&D.
- Tracking and geometry kernel codes developed independently in Japan and CERN were merged successfully just in one day during the workshop in September 1995.

→ *We convinced ourselves that we were in the right direction.*

## ■ Bench mark executed

- Comparison of G3 and G4 minimal driver (geometry)
- Geantino full comparison (geometry + tracking)
- Muon events comparison (geometry + tracking + piim)
- Demonstrated that G4 is faster than G3

→ *These provided a convincing indication to R&D reviewers that GEANT4 would be faster than GEANT3.*



# Toward the R&D Final Goal

---

## ■ Geant4 Beta Version

- Released in July 1998
- Major characteristics of the release.
  - ◆ Open beta-version of the full Geant4 system together with documentation, examples and tutorials.
  - ◆ Provided the concept that STEP compliant CAD models can be directly used for physics simulation
  - ◆ Extended further the physics validation range, and provided low energy neutron transfer for radiation studies.
  - ◆ Optimized and evaluated the physics and speed performance of the system

## ■ Geant4.0.0 Production Release

- Released in December 1998
- Major characteristics of the release
  - ◆ The first full production version with detailed documentation, examples and tutorials.

## ■ *R&D (RD44) closed at the end of 1998*

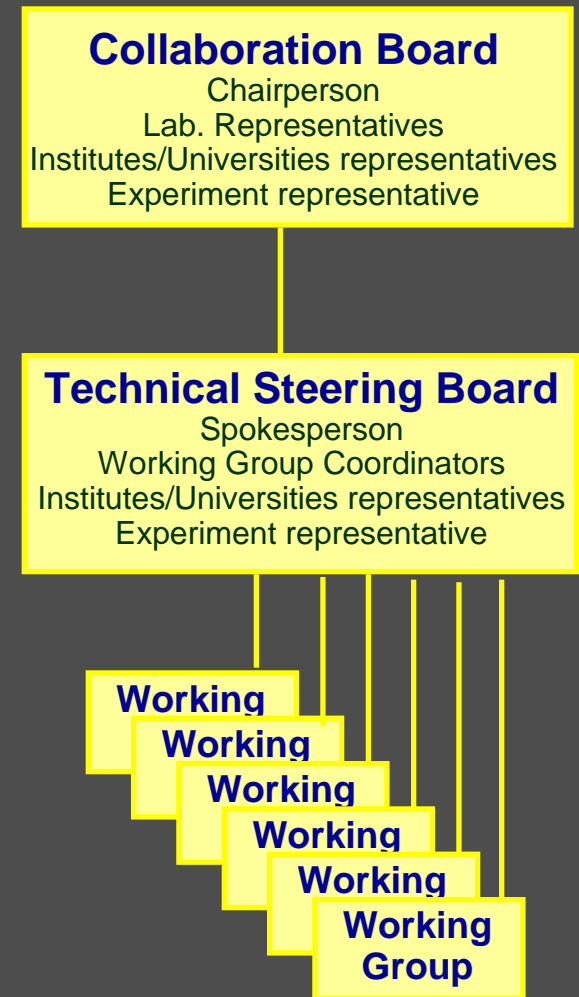
# Birth of Geant4 Collaboration

## ■ Organization

- A new scheme of the international collaboration named '*Geant4 Collaboration*' started immediately after the close of R&D - January 1999
- It is based on the Memorandum of Understanding (MoU) signed by laboratories, institutes and experiment groups over the world.

## ■ MoU

- To define a distribution of management, support and development of G4 software.
- The parties signed at the kick-off:
  - ◆ CERN, ESA, KEK, SLAC, TRIUMP, INFN, LEBEDV, LPNHE, ATLAS , BaBar, CMS, LHCb
  - ◆ More in the later stage



# Geant4 Code Release History

---

## ■ Basic policy of release

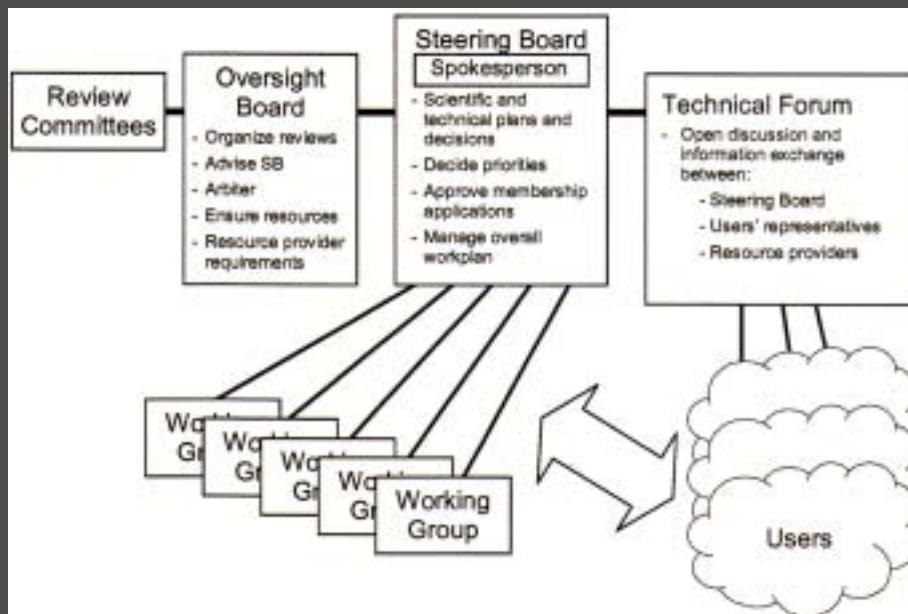
- Two to three public releases per year
- Monthly development tags for collaboration users

## ■ History

- 1998: G4\_0.0 (*Final product by R&D- RD44*)
- 1999: G4\_0.1 (*1<sup>st</sup> release in the G4 Collab. Phase*), G4\_1.0
- 2000: G4\_2.0, G4\_3.0
- 2001: G4\_3.1, G4\_3.2, G4\_4.0
- 2002: G4\_4.1, G4\_5.0
- 2003: G4\_5.1, G4\_5.2, G4\_6.0
- 2004: G4\_6.1, G4\_6.2, G4\_7.0
- 2005: G4\_7.1, G4\_8.0
- 2006: G4\_8.1, G4\_8.2
- 2007: G4\_8.3, G4\_9.0, G4\_9.1

# New Organization Structure and License

- New organization structure created in 2005



- Geant4 Software License Created

- 28 June, 2006: Version 1.0
- <http://geant4.web.cern.ch/geant4/license/LICENSE.html>

# Summary

---

- The history of Geant4 goes back to the two R&Ds started independently at CERN and KEK on object-oriented software development for LHC.
  - Geant4 is 14 years old this year!
- Geant4 was designed fully applying the OO methodology.
- Geant4 was designed as a toolkit and not an all-in-one and ready-to-use system. This allows you to integrate your own physics process by yourselves. Also it enables you to customize it for your applications - though with your own risk.
- In its implementation industrial standard tools were utilized for most cases. Also exploited were algorithmic know-hows accumulated in GEANT3.
- The Geant4 project is the first example of successful development of a large scale software in world-wide collaboration in the HEP community.
- Through the R&D and 'Geant4 Collaboration' phases Geant4 has grown to a product used not only in HEP but also in space, radiation, nuclear, medical... applications.