ELSEVIER

Nuclear Instruments and Methods in Physics Research A 453 (2000) 455–460

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

www.elsevier.nl/locate/nima

Present status of Geant4

Katsuya Amako

On behalf of the Geant4 Collaboration

High Energy Accelerator Research Organization, KEK, Institute of Particle and Nuclear Studies, 1-1 Oho, Tsukuba-shi, Ibaraki-ken 305, Japan

Accepted 21 June 2000

Abstract

Geant4 is a software toolkit for both full and fast Monte Carlo simulation of detectors in high-energy physics experiments. It has been developed under a worldwide collaboration of about 100 scientists, coming from over 40 institutes and HEP experiment groups in 15 countries. The first production version of Geant4 was released in December 1998. In this report, the present status of the toolkit is summarized.

1. Introduction

Geant4 is a general-purpose simulation toolkit for particle detectors. It provides basic functionality of simulation as to describe detector geometry and materials, to transport particle, to describe detector response and to visualize simulation related information. It also provides extensive physics models to describe interactions of particles with matter across a wide energy range. It was designed and developed under the CERN R&D project (RD44) [1–4] from the end of 1994 through 1998. RD44 was a worldwide collaboration of about 100 scientists, coming from over 40 institutes and experiments in 15 countries. The design goal of Geant4 was to create a flexible and extensible simulation toolkit exploiting object-oriented methodologies and C++ implementation technology. Achievement of transparency of the physics implementation and hence the possibility of validating the physics results with available experiment data was another important goal. To realize these goals, we accumulated basic user requirements from the HEP community. Also taken into account were requirements from space and cosmic ray applications, nuclear and radiation computations, heavy ion and medical applications. RD44 finished its mandate by releasing the first production version in December of 1998. After this release, major HEP laboratories and experiments over the world formed a new international collaboration – Geant4, which is based on the Memorandum of Understanding (MoU). This new Geant4 collaboration has a responsibility to maintain the production phase of the toolkit.

2. Geant4 development and software methodology

Because of wide variety of requirements to the Geant4 toolkit not only from the HEP community but also from other fields, the final product was expected to be a large and complex software system. It was also envisaged during the stage of

0168-9002/00/$ - see front matter
PII: S0168-9002(00)00681-1

planning the project that its construction would require HEP software expertise dispersed all over the world. This led to the formation of a worldwide development team.

To develop a complex software system like Geant4 in a worldwide collaboration, it is absolutely essential to employ an engineering discipline. The Geant4 collaboration studied feasibility of various object-oriented methodologies during the first year of the project to introduce an engineering discipline for the development. The requirements to a software methodology for the Geant4 development were: (1) it should have a flexible process, (2) it should provide a way to divide a system into independent subsystems so that a clear job sharing scheme (not only for code implementation, but also for analysis and design) could be defined, (3) it should provide models, notations and tools which help to exchange design ideas among people dispersed all over the world, (4) it should provide an incremental development process, and (5) it should provide a reverse engineering capability to guarantee a way to follow the evolution of the methodology.

The project decided to employ the Booch methodology [5] because it satisfied not only the requirements described above but also had a pragmatic and commonsense approach with an incremental and iterative process. The methodology had easy-to-understand models with rich notations which filled the gap between design and implementation and had a easy-to-obtain supporting software on Unix and PC environment. Although we selected the Booch method, we did not blindly adopt it. We adapted the methodology by taking into account HEP specific software environment.

In Booch methods, a software development is structured into micro and macro processes. The micro process serves as the framework for an iterative and incremental approach of the development. It is similar to the so-called spiral model. This process consists of a spiral loop of four sequences: (1) to identify classes and objects, (2) to identify class and object semantics, (3) to identify class and object relationships, and (4) to specify class and object interfaces and implementation. On the other hand, the macro process serves as the control framework for the micro process. It is close to the waterfall model. This process consists of five phases: (1) requirement, (2) OO analysis, (3) OO design, (4) evolution and (5) maintenance.

In the following section we summarize activities done in each phase of the macro process. The various aspects we found important in each phase for the worldwide collaboration are also described.

2.1. *Object-oriented analysis* (*OOA*) *phase*

We started this phase by collecting user requirements for the Geant4 detector simulation tool kit. he Geant4 core team (see below) prepared a draft of requirement document and it was distributed among the GEANT3 user community. Taken into account feedback from the community, the requirements were summarized and formatted in the ESA standard format [6].

Then we analyzed the requirement document to identify all major classes in the problem domain, including all data attributes and major operations that would be necessary to carry out functionality of the Geant4 toolkit. We produced central models (class diagrams) containing all the semantics of the toolkit in a set of concise but detailed definitions. We also identified clusters of classes (class categories) that were themselves cohesive, but were loosely coupled relative to other clusters. Major design documents produced in this phase were user requirements, class diagrams, scenario (object interaction) diagrams and a class category diagram.

In this phase the core team with a relatively small number (6–7) of people played an essential role. Since the number of people was small, the members could work closely together even if they were from various countries – Switzerland, England, France and Japan. From the beginning the Geant4 project was a truly worldwide collaboration.

2.2. *Object-oriented design* (*OOD*) *phase*

The goal of this phase was to elaborate the models created during the analysis phase so that the classes and objects could be coded and executed. Major design documents produced in this phase were updated class diagrams and scenario (object interaction) diagrams. We found that OOA and

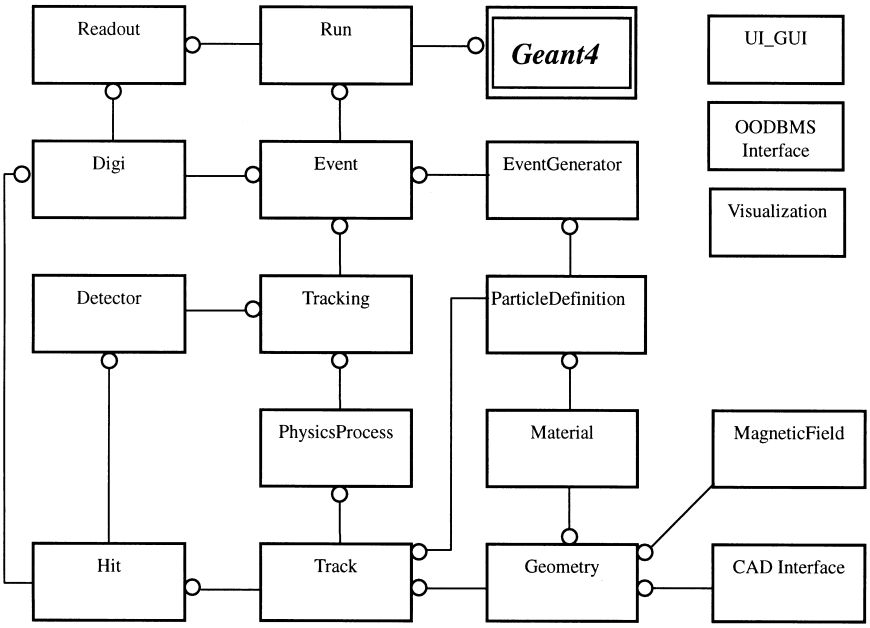

Fig. 1. Class category diagram of Geant4.

OOD were concurrent processes during the design of most class categories. About two-thirds of the first year was spent only for OOA and OOD, i.e. no intensive C ++ code development was committed during this phase.

The class category diagram (Fig. 1) created at the end of OOA had a fundamental importance during this phases of the worldwide development. We used each class category as a unit to share tasks in the OOD phase. Cohesive and loosely coupled nature of a class category enabled us to work on a detail design of classes in it in relatively independent to other categories. Because of this a much larger number of people could work together under each category comparing to the OOA phase. The concept of class categorization was essential for efficient worldwide collaboration.

2.3. Evolution phase

We started this phase by implementing and prototyping the design produced in OOA and OOD using the C ++ programming language. Cohesive and loosely coupled nature of a class category again played an essential role in this phase – we could proceed with the implementation of classes in a class category independently of those in other categories. Further refinement of the design was done during the prototype coding. Regular incremental releases of prototype codes and progress reports to the CERN R&D committee provided clear milestones for the project. Each release of prototype codes gave us chances to test regularly the quality of the product. By this iterative development, we could avoid a so-called big bang integration – to integrate all software components at once at the final stage of the project. The iterative nature of the micro process in Booch methodology played an essential role in the development during the evolution phase.

3. Overview of the Geant4 functionality

3.1. Run and Event

The concept of "run" is basically the same as in a HEP experiment. It keeps a set of events to be simulated using the same detector geometry, the same event-generator and the same physics

processes. Geant4 can execute many runs in one simulation program. An object of the class G4Run represents a run.

An "event" has a collection of primary particles to be tracked. Its concept is the same as an event in HEP experiments. An object of the class G4Event represents one event. The class G4EventManager controls the simulation of behavior of all particles that belong to an event. During the simulation of an event, one can use a triple prioritized stacking mechanism for tracking particles. The user can control the priority of tracking a particle by this mechanism. This is useful to study for pile-up, trigger, and loopers. An object of the class G4Event keeps results of simulation, for example, collections of trajectories, hits along tracks and digitization results.

3.2. Tracking

The tracking category manages the propagation of a particle though the detector taking into account its physics interactions with matter. An object of the class G4TrackingManager controls the propagation of a particle in the detector. The concept of "track" represents physics information (position, momentum, etc) of the particle under propagation. An object of the class G4Track describes a track. The G4TrackingManager propagates a particle in step-wise manner by delegating the control of one step to the class G4SteppingManager. The concept of "step" represents physics information of the particle during one step. An object of the class G4Step describes a step.

3.3. Particle

Geant4 provides a set of ordinary particles such as electron, proton, gamma, etc. It also covers more than 400 elementary particles.The class G4 ParticleDefinition provides a PDG compliant representation of a particle. Each particle has its own class derived from it. The class G4 ParticleDefinition contains information of particle properties which characterize individual particles such as name, mass, charge, spin, etc. To control the propagation of particles, Geant4 does not utilize the so-called tracking-cuts. Instead it uses the production thresholds, which are given in range, for secondary particles. Once a particle is produced it will be tracked up to zero range. This ensures an overall coherence of the simulation and enables Geant4 to exploit fully the validity range of physics interaction models.

3.4. Geometry

The Geant4 geometry provides three ISO STEP [7] compliant types of solid model – constructive solid geometry (CGS), boundary representation (B-rep) and swept area solid (SWEPT). Following the standard allows the Geant4 user to exchange geometry information between a detector simulator and a CAD system. Geant4 also has functionality of boolean operations to solids. Parameterizations in solid shape and material data are also available.

The class G4Navigator has the equation of motion solvers with geometrical boundary conditions for the propagation of particles in different fields, such as magnetic field, electric field and mixture of them. To improve tracking performance, G4Navigator utilizes the technique called "smart voxel". This new technique was developed from the idea of virtual division of a volume in GEANT3.21 and the voxel method, in which space is recursively subdivided into octants. The technique provides an automatic optimization to search volumes efficiently.

3.5. Physics processes

A sophisticated object-oriented design is employed to decouple implementations of tracking and physics interactions, which enables the user to integrate one's own physics process without the knowledge of the Geant4 tracking algorithm.

The way cross-sections are input or computed is split from the way they are used or accessed. The user can overload both these features. The way the final state is computed can again be split into alternative or complementary models, according to, for example, the energy range, the particle type and material. Multiple implementations of physics processes and models are possible and available.

The electromagnetic physics manages lepton, gamma and optical photon, as well as the electromagnetic interactions of muons, hadrons and ions.

The hadronic physics offers both a data- and parameterization-driven set of models, and a variety of theory-driven models for physics beyond test-beams energies, as well as treating low-energy neutron transport.

3.6. Fast simulation

The Geant4 framework is built to allow "Fast Monte Carlo" simulation that co-work with the full-event simulation. This can be done, for example, by plugging-in a user's shower parameterizations. The fast simulation allows a user-written process to take control from the full simulation and to parameterize the results of a shower directly into produced hits. It can also return tracks that can be passed to full simulation to continue more detail tracking. This ability can be particularly useful, for example, in a detector region with cracks. A parameterization can be triggered by particle type, by volume or a combination of these or other factors. The fast simulation can also occur transparently in a dedicated geometrical description independent of the one used by the tracking.

3.7. Hit and Digi

The classes G4Hit and G4Digi provide the functionality to reproduce the read-out structure of the detector and its electronic response, independently of the geometry used for the tracking. G4Hit provides a way of saving a snapshot of what happened in a certain region of the detector at a certain moment during the particle propagation. G4Digi keeps information of the actual detector response, which has to be created with an additional level of simulation.

3.8. Visualization and user interface

The visualization and GUI design is based on an object-oriented abstract interface that makes Geant4 independent from any particular graphics system. At the same time, this interface allows multiple implementations of drivers for graphics libraries, such as OpenGL, Inventor, DAWN (for engineering drawings) and VRML. In terms of user interfaces, it offers a batch or command-line approaches, and also modern graphical user interfaces (OPACS or Momo).

3.9. Persistency

Persistency provides a way to store Geant4 objects. An object database system or another solution can be used to realize this. Capability of persistency has been designed, developed and demonstrated. This includes storing and retrieving data of events, hits and the geometrical description of a detector.

4. Present status

4.1. Brief history

In 1993 the GEANT team at CERN investigated the class hierarchy structure in the geometry description employed in GEANT3.21 using the C++ programming language. In the same year, people from KEK and several universities in Japan studied an object-oriented analysis and design of the GEANT program. Their result was presented at CHEP94 [8]. In the fall of 1994, these two activities were merged and about 30 scientists from the world submitted an R&D proposal to construct an object-oriented toolkit for simulation in HEP to CERN/DRDC [1]. This project (RD44) was approved at the end of 1994, with the milestones for the first year to produce an object-oriented design of new simulation toolkit and a prototype to study its performance.

The project was re-approved at the end of 1995 for the following 18 months. The major milestone set by CERN/LCRB in this period was to realize an alpha-version of the product reproducing the basic functionality of GEANT3.21 in terms of geometry, tracking and physics. The alpha version was successfully released by late spring of 1997. The status report was submitted to CERN/LHCC/LCB in June 1997 [3], and the project was re-approved.

In 1998 the public beta version was released in July, and the first production version was made available in December of the same year.

4.2. Codes and environments

The production version of Geant4 consists of about 1 million of C++ code and about 2000 classes. The code runs on various workstation environments like IBM-AIX, DEC-alpha, HP, Sun and SGI. The user can choose either a native compiler or g++ on these machines. Also supported are Linux with g++ and Windows-NT with Visual C++. No commercial software is needed to run Geant4. Free software necessary to run Geant4 are CVS, gmake, g++ with STL and CLHEP library.

4.3. Documentation and code examples

Documentation and a set of example programs are available for users of various level. The documentation is structured with clear differentiation between the user and reference guides. The user guide consists of two volumes: one is for application developers and other for toolkit developers. Also available are an introduction and an installation guides. The reference guide consists of software reference manual and physics reference manual.

Six examples have been created to demonstrate the essential capabilities of the Geant4 toolkit to a novice user. These examples show the user how to use correctly the toolkit by implementing in a correct manner the user-classes which the user is supposed to customize in order to define the user's-specific simulation setup.

4.4. Performance and experience

To study performance of Geant4, various benchmark tests have been done. These include careful comparisons of performance between Geant4 and Geant3.21 using the standard test geometry of the liquid argon and lead sandwich calorimeter. The result showed that, at a comparable physics performance, Geant4 is faster than Geant3.21 by a factor of more than three using exactly the same cuts. Also by a factor of more than 10 faster when Geant4 cuts are optimized, keeping the same physics performance. More details of these studies can be found in the "Geant4 Gallery" under the Geant4 official web page [9].

Our studies show that many factors contribute to these results. These include optimized design, more powerful geometry navigation and tracking algorithms, improved physics algorithms, etc.

Geant4 has been already adopted by various large-scale experiments for example ATLAS, BaBar, CMS and LHCb. It is also utilized in other research fields like radiation background studies at LHC, space application at ESA (European Space Agency) and medical application.

References

[1] A. Dellacqua et al., Geant4: an object-oriented toolkit for simulation in HEP, CERN/DRDC/94-29, August 1994.
[2] G. Cosmo et al., Geant4: an object-oriented toolkit for simulation in HEP, LCRB Status Report, CERN/LHCC/95-70, October 1995.
[3] T. Wenaus et al., Geant4: an object-oriented toolkit for simulation in HEP, LCB Status Report, CERN/LHCC/97-40, June 1997.
[4] S. Giani et al., Geant4: an object-oriented toolkit for simulation in HEP, LCB Status Report, CERN/LHCC/98-44,November 1998.
[5] G. Booch, Object-Oriented Analysis and Design with Application, 2nd Edition, The Benjamin/Cummings, Menlo Park, CA, 1994.
[6] ESA Software Engineering Standards, European Space Agency, ESA PSS-05-0 Issue2.
[7] ISO STEP: ISO 10303-42, Integrated Generic Resources: Geometric and Topological Representation.
[8] K. Amako, et al, Object-oriented analysis and design of a GEANT based detector simulator, Proceedings of CHEP94, San Francisco, CA, USA, LBL-35822 CONF-940492.
[9] The Geant4 Collaboration, http://wwwinfo.cern.ch/asd/geant4/geant4.html.