Mantis: the Geant4-based simulation specialization of the CMS COBRA framework

M. Stavrianakou, *FNAL, Chicago, USA*, P. Arce, *CIEMAT, Madrid, Spain*,
S. Banerjee, *Tata Institute, Bombay, India*, T. Boccali, *SNS, Pisa, Italy*,
A. De Roeck, V. Innocente, M. Liendl, *CERN, Geneva, Switzerland*,
T. Todorov, *IReS, Strasbourg, France*

Abstract

Mantis is a framework and toolkit for Geant4-based simulation using the CMS OO architecture implemented in the COBRA project. Mantis provides the infrastructure for the selection, implementation, configuration and tuning of all essential simulation elements: detector geometry, sensitive detectors, generators and physics, magnetic field, run and event management, and user monitoring actions. Persistency and other important services are available using the standard COBRA infrastructure and are transparent to user applications.

INTRODUCTION

Mantis is a framework and toolkit for Geant4-based [1] simulation. As such, it is a specialization of the COBRA framework [2], which implements the CMS OO architecture. In CMS, Mantis is the basis for the OSCAR [3] simulation program.

ARCHITECTURE, REQUIREMENTS AND DESIGN

Mantis has been developed in the context of COBRA. COBRA is a mature product based on state-of-the-art design patterns and implementations. It has been proven flexible and extensible, reliable, robust, performant and maintainable.

Mantis is architecture-driven and has from its conception aimed to exploit fully the COBRA infrastructure and functionality. Several COBRA choices (lazy instantiation, action on demand etc) and patterns (factories, observers-dispatchers etc) are indeed applicable to the simulation domain and have been used extensively. Given the vast scope of LHC simulation, extensibility and configurability are fundamental prerequisites: abstract interfaces allow implementation of new modules for all major simulation components. Although most essential functional requirements are taken into account from the start, further requirements are implementable with minimal effort.

To minimize overheads from development and support of non simulation-specific elements and to facilitate maintenance and quality assurance, Mantis has also aimed at maximal synergy with other CMS software products and tools. For visualization and event display Mantis relies on IGUANACMS [4]. Configuration management and the release process are managed with SCRAM [5]. Important quality assurance functions such as dependency analysis, profiling and testing are handled by CMS IgNominy, IgProf and OVAL [5] respectively.

Important constraints arise from demands on ease of use. This implies minimal need for coding, compilation and linking on the user side, as well as run-time flexibility and configurability.

Requirements and constraints also arise from the need for simulation productions. Metadata initialization and the capability to suspend and resume interrupted runs (the latter being especially important for Condor farms), are examples of such requirements. In general, Mantis is compatible with the available infrastructure and has required minimal additional deployment effort by the CMS Monte Carlo production team.

Figure 1: Mantis domain decomposition and dependencies

A schematic Mantis domain decomposition and the resulting dependencies is shown in Fig. 1. This reflects the current implementation (which amounts to less than 10 KLOC) that is already the result of several iterations of domain breakdown and re-engineering. The very loose coupling (see metrics in Table 1), explained in great detail in [6]) allows further re-engineering and repackaging, if desirable, in a manner transparent to the client applications.

Table 1: Mantis metrics calculated by IgNominy

Packages	10
Levels	5
Cumulative Component Dependency (CCD)	30.0
Average Component Dependency (ACD)	3.0
Normalized CCD (NCCD)	1.1

CORE APPLICATION, RUN AND EVENT MANAGEMENT

Mantis is a COBRA simulation application and relies on it for the program's *main*. In this manner, it is COBRA that takes over and handles the application. The core consists of a *SimApplication* and a *SimEvent*.

SimApplication creates the *SimReader*, an instance of a *SimEvent* source factory. The *SimEvent* source factory is a COBRA abstract factory (a statically built singleton) that can read simulated events from the original source.

The *SimReader* instantiates the Mantis *RunManager* and launches the actual Mantis/Geant4 application.

The *RunManager* instantiates a *G4RunManagerKernel* and controls standard components such as the selection and instantiation of generator, magnetic field and physics lists, as well as the interfaces to the run, event, stacking, tracking and stepping actions. The *RunManager* also handles the storage and retrieval of run and event random number seeds. This feature facilitates debugging of rare crashes in time-consuming physics events. The storage and retrieval of the cross-section tables built for a given detector configuration and physics list is also thus centrally handled. This feature allows the reading of pre-built cross-section tables which reduces the overall initialization time by approximately a factor 4. At the end of the program, control is returned to COBRA.

Although a Mantis application can be run interactively in a Geant4 shell, with or without a Geant4 macro command file, the recommended mechanism is via the *mantisRC* datacard file. This is the standard COBRA configuration mechanism and ensures maximum flexibility as well as overall consistency.

The event object is an instance of a COBRA *SimEvent*. A *SimEvent* manages the Monte Carlo truth, the format of which is common for and sharable by all CMS applications (reconstruction, visualization etc). It combines an interface to the generic transient event *TSimEvent* with an interface to the basic Geant4 event.

The Monte-Carlo truth is assembled by the Mantis *EventAction*. It contains the main event, its assigned weight and its type as specified by a set of parameters, the event ID (run number, event number), the four-vector of the collision vertex and all particles with their tracks and vertices and decay trees from the original generator event. Tracks produced during the Geant4 simulation are also stored if they have been flagged for saving at various points (tracking, stepping, hit processing etc) of the actual CMS simulation. Geant4 tracks are saved either because they have produced hits in the sensitive detectors or because they have been identified as important for the interaction history and the eventual reconstruction of the full tree.

The Monte-Carlo truth is organized so as to allow navigation from hits to their corresponding tracks and parent vertices.

A COBRA *SimDBPopulatorFactory* factory interfaces to the persistent store.

INFRASTRUCTURE AND SERVICES

Geometry

Detector description in CMS is handled by the Detector Description Database, DDD [5], a COBRA subsystem. The XML files with the actual detector configurations are under version control, managed by the Geometry project[5].

Mantis provides mechanisms to convert DDD solids and materials to their Geant4 counterparts as well as the logical and physical volumes needed to build the Geant4 geometry for the chosen description.

The DDD *SpecPars* mechanism allows the definition of special parameter sets (extra attributes, field parameters, range cuts etc) to be associated with selected detectors.

Magnetic Field

The magnetic field inherits from *G4MagneticField* and implements the Geant4 *GetFieldValue* method, so that any standard CMS magnetic field can be loaded by the COBRA *CMSMagneticFieldLoader* and passed to Geant4. It is a world volume observer, i.e. it can only instantiate a field when it is notified that the detector setup has been built. The DDD *SpecPars* mechanism allows choice and configuration of *G4MagIntegratorStepper* (*G4ClassicalRK4*, *G4SimpleHeum*, *G4HelixExplicitEuler* etc), chord-finder and propagator. The infrastructure also allows modeling, instantiation and configuration of local field managers for chosen detectors and particles.

Local field managers, an important Geant4 feature, handle particles that are either of little interest or unlikely to escape a given detector or set of volumes and can therefore be propagated with relaxed criteria as to the accuracy of the stepping and chord finding. This treatment allows a moderate performance improvement (propagation in field

accounts for about 10% of the processing time) depending on the type of study and particles involved . The obvious use case currently under study involves all particles other than muons or charged pions of $E_{kinetic} > E_{threshold}$, in all calorimeter volumes.

Sensitive Detectors

Sensitive detectors inherit from *G4VSensitiveDetector* and their concrete implementations in the actual CMS application implement the Geant4 *ProcessHits* method. The sensitive detectors are registered to the Geant4 sensitive detector manager but they are instantiated and "attached" to their corresponding geometrical volumes at run time according to the set-up described in the configuration DDD/XML file. The facility is a world volume observer, i.e. it must be notified that the geometrical detector has been constructed before it can be instrumented.

This mechanism also offers the possibility of instrumenting (making sensitive) any volume, for prototyping purposes or in order to facilitate studies of energy losses in dead materials or specific parts of the detector.

While all real CMS sensitive detectors are implemented in OSCAR, a general-purpose sensitive detector, given as a Mantis example, can be used for stand-alone Mantis testing.

Hits and hit collections

Hit processing and collection is handled outside the framework by the detectors themselves, based on COBRA classes common for and sharable by all CMS applications. These common classes are managed by the COBRA Profound package.

COBRA read-out factories handle the hit formatting as will be required for the digitization. The latter which is handled separately from the simulation, by the CMS ORCA [5] reconstruction program.

Generators

Event generators are constructed from an abstract factory based on the COBRA GeneratorInterface packages and services. The *RunManager instantiates* the chosen generator (also referred to as *reader*), if any. The generator *trigger* method returns a *RawHepEvent* or a *HepMC::GenEvent*, which is passed to the *RunManager GenerateEvent* to be converted to a *G4Event*. The conversion method (*RawHepEvent2G4* or *HepMC2G4*) creates a *G4PrimaryVertex* with the generated vertex coordinates and assigns to it the primary particles that survive acceptance cuts. This primary event vertex, the proper decay times of the unstable particles and their predefined decay products are passed to the *G4Event*. Thus it is Geant4 that creates the secondary vertices after correctly propagating and simulating the unstable particles until they decay as predefined.

All generators can be run-time configured in terms of

- the first event to be read (automatically determined in the case of interrupted run resumption)
- the maximum number of events they can return depending on the total available (or a sensible number in the case of on-the-fly generation) and the first event esp. if non-zero or if the run is being resumed
- the event vertex generator to be used (none, flat, Gaussian, test-beam specific)
- the run and event numbering scheme to apply

Available generators (some provided for backwards compatibility to allow reading of already available samples):

- *EventGunReader*: a particle gun with run-time choice and configuration of particle type, and ranges, energy and pT ranges with a given distribution (flat, Gaussian etc)
- *EventNtplReader* and *EventTxtReader*: read CMS generated physics events from HBOOK ntuple or ASCII file
- *EventPythiaReader*: read a *Pythia6* event generated on the fly, using the COBRA *Pythia6Interface*
- *EventStdHepReader*: read an event in *StdHep* format from an ASCII file
- *EventHepMCReader*: read a *HepMC::GenEvent* from any type of input file (ASCII, POOL database, on-the-fly etc)

Physics

An abstract physics list factory allows run-time selection and configuration of specific physics lists. Physics cuts (i.e. range cuts) are implemented as cuts per region (set of volumes). The regions and volumes they contain (typical scheme distinguishes between "sensitive" and "dead" regions) and the cut values for electrons, positrons and photons are read from a DDD/XML file at run-time.

The CMS physics lists reside in and are managed by OSCAR. A *DummyPhysics* list in Mantis, with propagation, decay and optionally *DummyEMPhysics* (only ionization process for electrons and muons) is provided for Mantis stand-alone testing.

User Actions

The Geant4 *UserAction* mechanisms are employed to dispatch (using the COBRA *Dispatcher-Observer* pattern) quantities such as the beginning and end of run, event, track and step.

User monitoring is implemented in the form of Observers of one or more of these quantities with access to the dispatched pointer via the COBRA *Observer::upDate* method

Miscellaneous services

Persistency is handled by COBRA with Mantis housing minimal and stable interfaces to it. This scheme has allowed transparent transition from Objectivity/DB to ROOT and eventually to POOL, without any change in Mantis or user code.

Similarly, reliance on COBRA for histogramming and statistical analysis services has facilitated the transition from HBOOK and LHC++/Anaphe, to AIDA and ROOT.

As mentioned previously, monitoring, including production monitoring features and GRID interfaces, and software configuration management and quality assurance, are all handled by external to Mantis tools.

New functionality

Mantis is supposed to provide a framework for partial event simulation for CPU-intensive very large scale productions of physics channels with appropriate commonalities, such as the $H \rightarrow ZZ \rightarrow 4$ leptons, for which 50 M event samples must be simulated. The ability to simulate the event without the leptons and then superimpose the leptons, which are simulated separately but with the correct kinematics, will significantly reduce production timescales and costs.

Although Mantis is by design Geant4-based, an interface to FLUKA physics using the same Geant4 geometry via the FLUGG package, can be accommodated. Access to FLUKA physics may be required for specific detector studies of radiation effects on sensitive systems.

Should it emerge as a CMS requirement, Mantis can be extended to provide interfaces for all CMS simulation applications, allowing combinations of and transitions between fast simulation with FAMOS [5] and full simulation with OSCAR [3] with or without parameterized shower simulation with G4FLASH [7] and with or without FLUKA interfaces.

MANTIS APPLICATIONS: OSCAR

Mantis is the framework underlying the CMS Geant4 simulation program OSCAR [3], [7].

All CMS detectors - Tracker, Calorimeters and Muons, the forward systems - CASTOR calorimeter, Totem telescopes and recently the Zero Degree Calorimeter (ZDC), and several test beam layouts and prototypes, are built using the Mantis/DDD/Geometry infrastructure. Their sensitive detector behaviour, track selection mechanisms, hit collections and numbering schemes are implemented in OSCAR.

OSCAR also provides several physics lists: the standard Geant4 with choice of hadronic list (LHEP, QGSP, QGSC and FTFP), a fully customizable list and recently a list for parametrized electromagnetic shower simulation.

The standard OSCAR application (Tracker, Calorimeters and Muons and the QGSP physics list) has been extensively tested following the evolution of Geant4 and Mantis. It has proven robust: 1/10000 crashes in pp events in the DC04 production to no crashes in the latest stress tests with 800 K single particles and 300 K full QCD events. It has produced over 35 million physics events for the CMS DC04 data challenge and is in use for the CMS physics TDR simulations. A SUSY event with leptons and missing transverse energy, as simulated with OSCAR and displayed by IGUANACMS, is shown in Fig. 2.

As of September 2004, the performance in terms of CPU and memory compares favourably to that of the Geant3-based simulation, with further improvements in the pipeline.

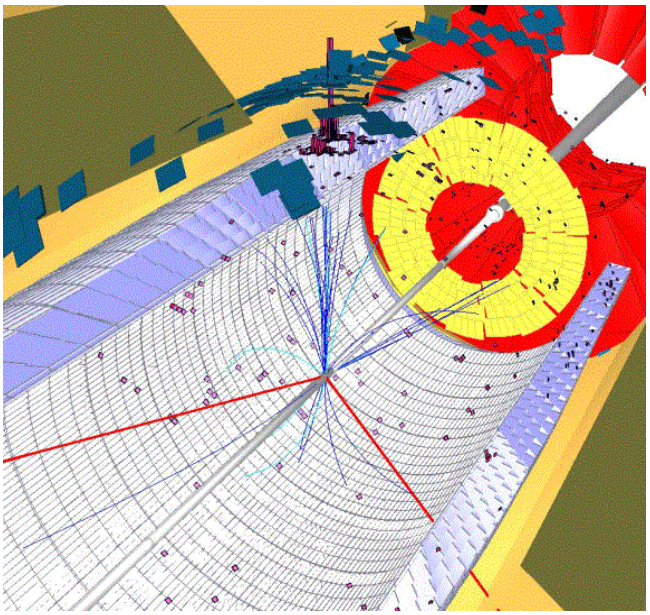

Figure 2: SUSY event simulated with OSCAR and displayed by IGUANACMS

REFERENCES

[1] S. Agostinelli et al., "Geant4: a simulation toolkit", NIM A 506 (2003), 250-303
Geant4, http://wwwinfo.cern.ch/asd/geant4/geant4.html

[2] V. Innocente, et al., "CMS Software Architecture: Software framework, services and persistency in high level trigger, reconstruction and analysis", Computer Physics Communications 140 (2001) 31-44

[3] OSCAR, Object oriented Simulation for Cms Analysis and Reconstruction, http://cmsdoc.cern.ch/oscar/

[4] V. Innocente, G. Eulisse, S. Muzaffar, I. Osborne, L. Taylor, L.A. Tuura, "Composite Framework for CMS Applications", CHEP'04, Interlaken, Switzerland, September 2004

[5] CMS Object-Oriented projects,
http://cmsdoc.cern.ch/cmsoo/cmsoo.html

[6] Large Scale C++ Software Design, John Lakos, Addison-Wesley, 1996

[7] "An Object-Oriented Simulation Program for CMS", S. Abdoullin et al., , CHEP'04, Interlaken, Switzerland, September 2004