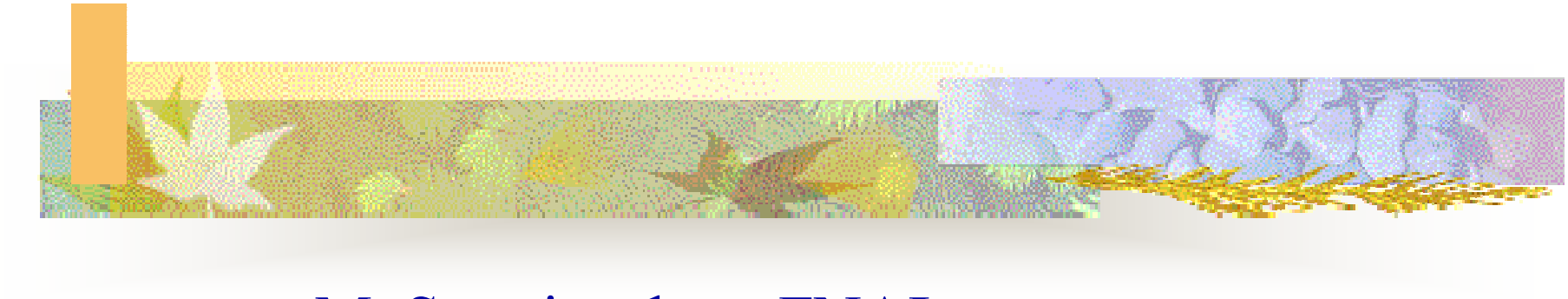
Mantis
the Geant4-based simulation specialization of the CMS COBRA framework



M. Stavrianakou, FNAL

M. Stavrianakou, P. Arce, S. Banerjee, T. Boccali, A. De Roeck, V. Innocente, M. Liendl, T. Todorov

Overview

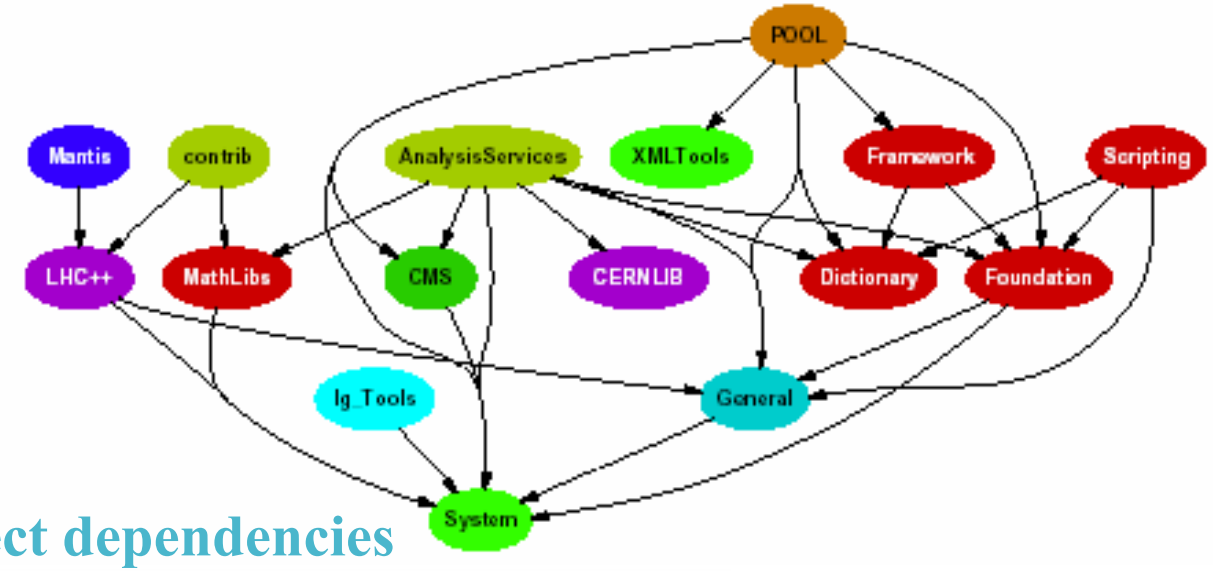
- Mantis = CMS Geant4-based Simulation Framework = specialization of the COBRA framework, which implements the CMS OO architecture; basis for the CMS-specific simulation program OSCAR
- provides the infrastructure for the selection, configuration and tuning of all essential simulation elements: geometry construction, sensitive detector and magnetic field management, event generation and Monte Carlo truth, physics, particle propagation and tracking, run and event management, and user monitoring actions
- experimental set-up built by Mantis using COBRA Detector Description Database, DDD, which allows transparent instantiation of any layout (full or partial CMS simulation, test beam set-ups etc)
- persistency, histogramming and other important services available using standard COBRA infrastructure; transparent to

Requirements, constraints and design choices

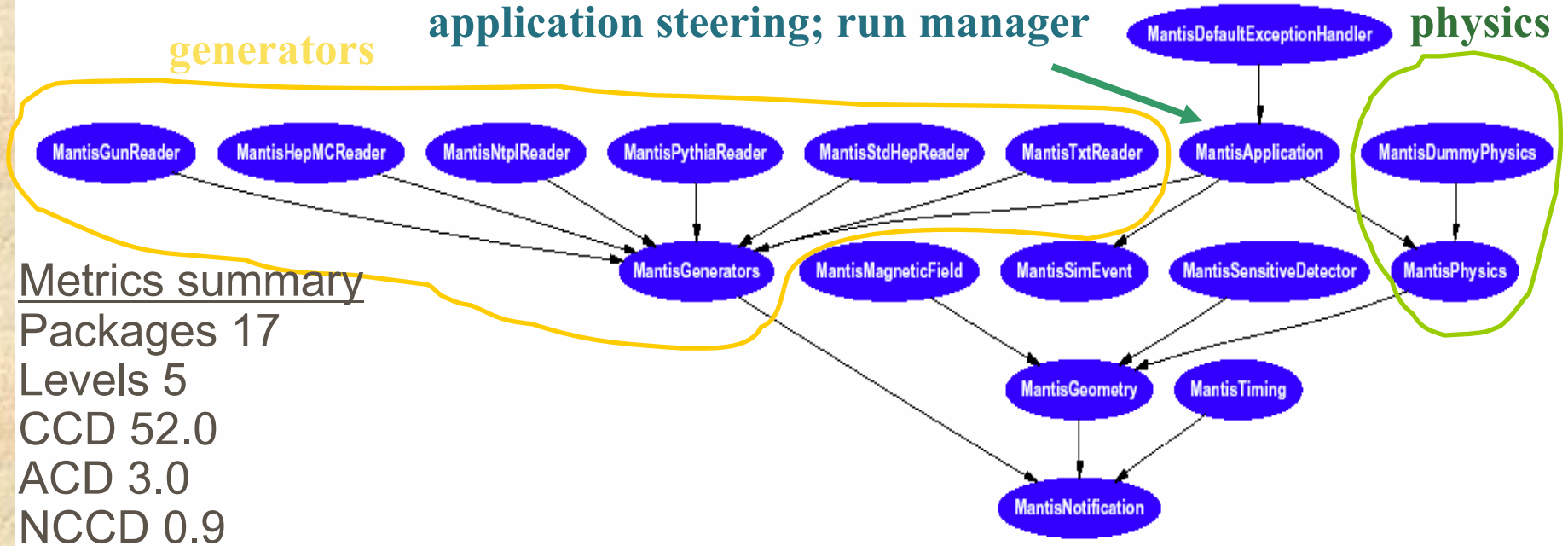
- architecture-driven approach in CMS COBRA framework
- modular design, maximal synergy with COBRA, minimal assumptions and constraints as to requirements of specific CMS simulation implementations
  - extensibility and configurability: abstract interfaces allowing implementation of new modules for all major simulation components (geometry modeling, generators, physics, user actions etc)
  - minimal need for coding, compilation and linking on user side, as well as run-time flexibility and configurability
  - development, testing and debugging of individual components without unnecessary overheads; e.g. geometry modeling and visualisation decoupled from physics simulation and event processing
  - maximal synergy with other CMS software products and tools for visualization (IGUANACMS), dependency analysis (IgNominy), testing (OVAL) etc
- support for simulation productions: compatibility with CMS infrastructure to ensure minimal deployment effort; incl. specific features such as
  - metadata initialization
  - capability to suspend and resume interrupted run

Dependencies and metrics

Mantis in COBRA



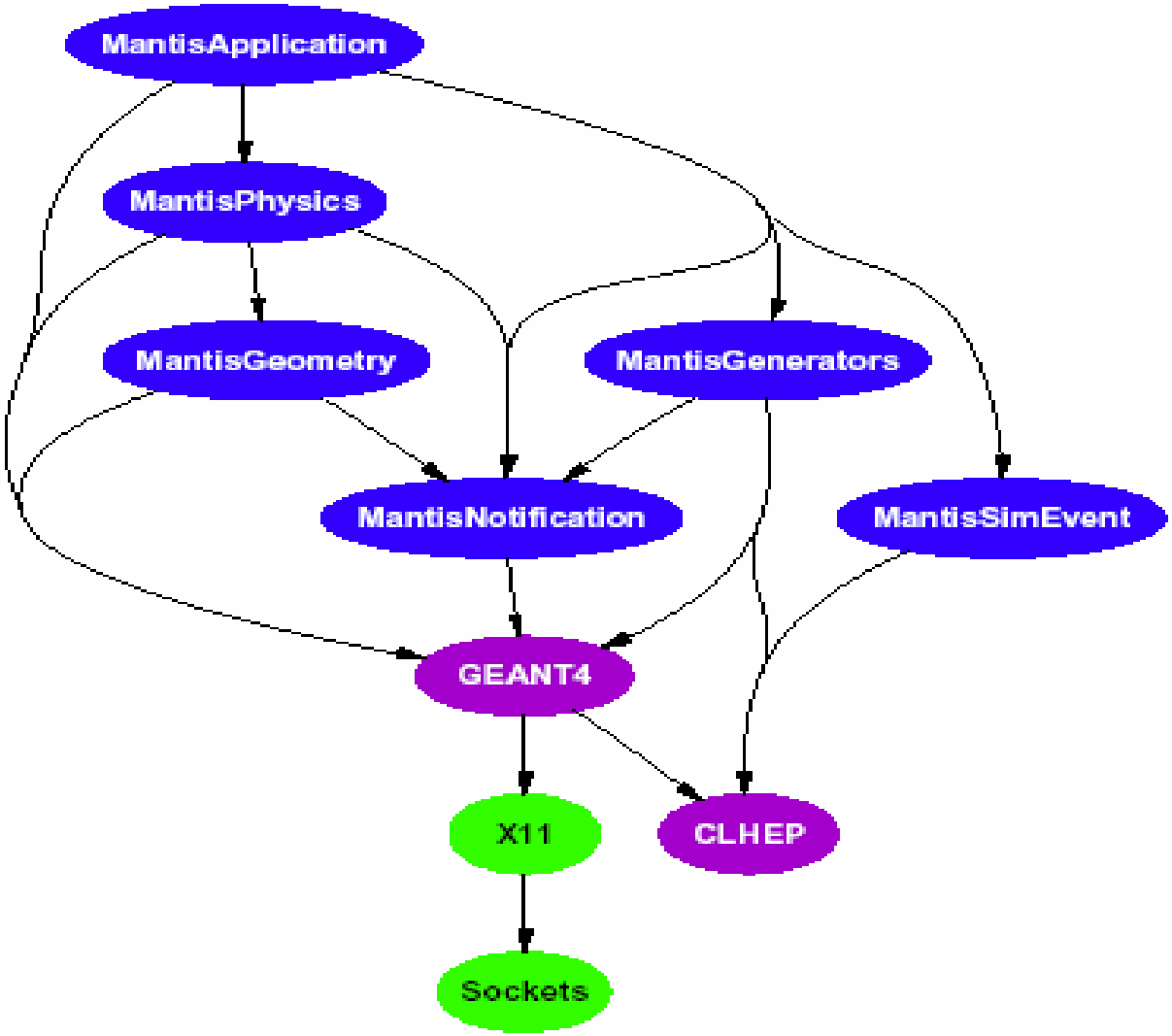
Mantis subsystem direct dependencies



Metrics summary
Packages 17
Levels 5
CCD 52.0
ACD 3.0
NCCD 0.9

Application steering and Run management

MantisApplication: detailed dependency tree



Application and Run management: the core

- relies on COBRA for program's *main;* COBRA takes over and handles the application; it consists of a *SimApplication* and a *SimEvent*
- the *SimApplication* instantiates and launches an event source factory (COBRA abstract factory - statically built singleton) - an abstract reader of simulated events from the original source; it instantiates the *RunManager*
- the *RunManager* instantiates a *G4RunManagerKernel* and controls standard components such as the selection and instantiation of generator, magnetic field and physics lists and interfaces to the run, event, stacking, tracking and stepping actions
- the *SimEvent*, inheriting from the COBRA *SimEvent*, manages the Monte Carlo truth - common for and sharable by all CMS applications (reconstruction, visualization etc); it consists of an interface to the transient event. (be it Geant3, Geant4, fast simulation) and an interface to a Geant4 event.
- hit processing, processing and collection is handled outside the framework by the detectors themselves based on COBRA classes common for and sharable by all CMS applications

Infrastructure and Services (I)

- **Geometry**
  - detector description in CMS is handled in the CMS Detector Description Database (DDD)
  - 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
- **Sensitive detectors**
  - sensitive detectors services are based on COBRA read-out factories allowing the registration and assignment of sensitive detectors at run time according to the set-up described in a configuration XML file
  - 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

Infrastructure and Services (II)

Generators

- an abstract generator factory based on the COBRA GeneratorInterface packages and services provides a *trigger* method to produce and return a *RawHepEvent*
- the RunManager instantiates the chosen generator (also referred to as “reader”), if any
- Some “standard” generators
  - EventGunReader: a particle gun with run-time choice of vertex generator (none, flat or Gaussian) and configuration (particle type, η and φ ranges, energy and p_T ranges with flat or Gaussian distributions)
  - EventNtplReader and EventTxtReader: read CMS generated physics events from ntuple or text file
  - EventPythiaReader: read a Pythia6 event generated on the fly, using the COBRA Pythia6Interface

Infrastructure and Services (III)

Physics

- abstract physics list allowing run-time selection and configuration, also in terms of physics cuts (i.e. range cuts) per region (set of volumes) described in XML

Magnetic field

- inherits from *G4MagneticField* and implements *GetFieldValue*, allowing standard COBRA choice of magnetic field, loaded by the COBRA *CMSMagneticFieldLoader*
- allows choice and configuration of G4MagIntegratorStepper (G4ClassicalRK4, G4SimpleHeum, G4SimpleRunge, G4HelixImplicitEuler etc), chord-finder and propagator
- allows modeling and instantiation of local field managers for chosen detectors and particles

Infrastructure and Services (IV)

Observables and user actions

- the Geant4 mechanisms for the so-called UserActions are employed to dispatch (using the COBRA implementation of the Dispatcher-Observer pattern) pointers to 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

What Mantis gets for free… Be lazy…

- Application control and interfaces to common services as described previously
- Persistency
  - has allowed transparent transition from Objectivity/DB to ROOT to POOL
- Histogramming and statistical analysis utilities
  - has allowed transparent transition from HBOOK to Anaphe, to AIDA and ROOT
- Monitoring
  - including production and GRID interfaces
- Geometry descriptions from DDD and Geometry
- Visualization from IGUANACMS
- S/W configuration management and quality assurance

Is Mantis really extensible?

sure seems so…

so far experience with

- support for new types of physics lists
  - recent addition of a parameterized e/m shower simulation (G4Flash)
- support for new generator input formats and sources
  - migration from CMS "RawHepEvent" to HepMC
    - with input from ASCII, POOL etc
- support for new magnetic field and local field managers
- straightforward adoption of common CMS tools and compliance with CMS standards
- continuous evolution following and in close collaboration with Geant4
- success of the CMS OO simulation *OSCAR*, based on Mantis

Configuration and datacards examples

In the card file

GoPersistent = true

ExtraPackages = MyPhysics:MantisNtplReader:MyExceptionHandler

RunManager:RestorePhysicsTables = true

DDDConfigFile = OSCARconfiguration.xml

EventNtplReader:NtplFileName = minbias.ntpl

Generator:ApplyEtaCuts = true

Generator:MinEtaCut = -5.5

Generator:MaxEtaCut = 5.5

Generator:VertexGenerator = GaussianEventVertexGenerator

MantisMagneticField:UseMagneticField = true

MagneticField:Name = CMSIMField

NumberOfEventsToBeProcessed = 10

Configuration and datacards examples*(cont'd)*

In OSCARconfiguration.xml

```
<Include>
   <File name="CMS/CMSGeometry/cms.xml"   url="."/>
   <File name="ECal/ECalGeometry/ecal.xml"   url="."/>
   <File name="Materials/materials.xml"   url="."/>
   <File name="Rotations/rotations.xml"   url="."/>
   <File name="ECal/ECalSensDet/ecalsens.xml"   url="."/>
   <File name="ProductionCuts/EcalProdCuts.xml"   url="."/>
   <File name="PropagationInField/FieldParameters.xml"
url="."/>
</Include>
<Root fileName="cms.xml" logicalPartName="OCMS"/>
```

Summary and outlook

- So far so good!
- Next: partial event simulation
  - when you need 50M events per channel…
    - e.g. H→ZZ→4l: μμμμ, eeμμ, eeee
  - simulate event without leptons, superimpose leptons simulated with correct kinematics
- And then?
  - *FLUKA* interface via *FLUGG*?
  - common framework for all CMS simulations?
    - from fast (*FAMOS*) to parameterized (*G4FLASH*) to full simulation (*OSCAR*)…
  - interactivity
    - not happy with *G4Messenger* system…
    - Python? ROOT? Other?

In collaboration with COBRA and the CMS distributed analysis project (APROM)