CMS Experience with Adoption of the Community-supported DD4hep Toolkit

Carl Vuosalo
University of Wisconsin-Madison
Ianna Osborne
FNAL

Presented at CHEP 2019, The 24th International Conference on Computing in High Energy and Nuclear Physics on behalf of the CMS Collaboration
Outline

• Motivation to adopt DD4hep
• Migration process and scope
• Code migration techniques
• Migration challenges
• Good practices for migration
• Summary
Need for DD4hep

● The Compact Muon Solenoid (CMS) collaboration developed and maintained for many years its own custom detector description (DD) for detector geometry

● CMS DD disadvantages:

➤ It is a singleton that doesn't support multi-threading
  - CMS software uses advanced parallelization techniques for improved performance that are blocked by singletons
    • See CMS poster “Concurrent Conditions Access across Validity Intervals in CMSSW” by Chris Jones for parallelization example

➤ Old code that is costly to maintain and enhance
  - Accumulated defects and obsolete sections over many years
  - No easy path to adopt innovations and new technology
CMS Detector Geometry

2021 model built with DD4hep
Old CMS DD

Detector and Special parameters description

XML

Geometry Scenario

C++

Algorithmic DD description and positioning ~100 CMS plug-ins

SAX

Parser

Mediator Compact View

One DD geometry source for all applications:
- Simulation
- Fast simulation
- Reconstruction
- Alignment and Calibration
- Visualization

Expanded View

Filtered View

Build G4 Geometry, set field manager

G4 Geometry

Attach SD

Reco Geometry

Event Display

Legacy code based on Singletons, not thread-safe
Benefits of DD4hep

- Supports **multi-threading**
- Fully featured
- **Community-supported** toolkit
  - Widely used in HEP by CALICE, FCC, ILC, LHCb, etc.
  - Benefits from innovations and contributions from across HEP community
- Will continue to **evolve** with advancing technology
- Commitment for years of maintenance and enhancement
- Development based at **CERN**
- DD4hep team very **responsive** to users' needs
CMS Using DD4hep

Detector and Special parameters description

DOM

Event Display

DD Detector Views

XML

Parser

DD Detector TGeo

“Compact” View

Geometry Scenario

Spec Pars

G4 Geometry and numbering

Expanded View

C++

Algorithmic DD description and positioning

Gaudi plug-ins

Reco Geometry and numbering

Filtered View

DD Detector Views implemented to optimize geometry navigation and minimize downstream code changes

5 November 2019

Carl Vuosalo – University of Wisconsin-Madison
Additional Benefits of Migration

• Migration provides opportunity to **improve** code base
  ▶ Drop unused shapes, features, and obsolete code
  ▶ Fix previously undetected overlaps of geometric volumes
  ▶ Refine geometry and enhance testing and validation
• Motivates improvement of DD4hep to meet CMS requirements
• Builds expertise among developers doing migration
• Demonstrates value of community-supported software
• HEP community faces huge upcoming computing challenges like HL-LHC
  ▶ Will need to **pool efforts** to meet these challenges
Migration Timeline

2018
- Start evaluation January 2018
- Perform test migration of small package
- Evaluation completed December 2018
- Result: Test migration successful with tolerable performance

2019
- Start migration January 2019
- Validation in progress
- Completing migration around year-end 2019

2020
- Optimization, early 2020. Improve performance and remove legacy features
- Full event simulation and reconstruction performed solely with DD4hep
- Eliminate old DD, late 2020
Scope of Migration

- CMSSW, the CMS software system, has about 6.5 million lines of code
  - Mostly C++ code, some Python and XML
  - Only small fraction needs to be migrated
- Detector geometry used for event simulation and reconstruction
- Roughly 150,000 lines of C++/Python code require migration
  - Several hundred files
  - Not all lines of code have to be changed, but they must at least be reviewed
  - 1.5 million lines of XML detector geometry description
    - XML requires only minor fixes, no major changes
    - 61 C++ algorithms called from XML require migration
- Half dozen developers performing migration
Techniques for Migrating Code

- Evaluation phase
  - Separate package for migrated code
  - Leave old code untouched
- Migration phase combines various approaches
  - Put migrated code into mainline development branch ("integration build")
  - Parallel migrated versions of some files put in \texttt{dd4hep} directories or given names starting with \texttt{DD4hep_}
    - Python script loads desired version
  - Some sections of migrated code activated by \texttt{fromDD4hep} flag
  - Some classes templated to provide old and migrated versions
  - Try to balance:
    - Preserving old behavior for validation of migrated code
    - Minimizing code duplication
Integration of DD4hep

- DD4hep handled as external tool in CMSSW
  - DD4hep built by CMS build system
  - CMS keeps up with DD4hep releases
  - Recent issue: DD4hep revised its cmake configuration
    - Required CMSSW fix to build new version of DD4hep
- DD4hep uses Gaudi plug-in format
  - CMS has its own plug-in format
  - CMS added rule to build system to support Gaudi plug-ins
- CMSSW uses both dynamic and static libraries
  - DD4hep added support for static libraries
Migration Challenges

- DD4hep lacked seven special features required by CMS geometry code
  - These features include special shapes and use of a left-handed coordinate system
  - DD4hep team enhanced DD4hep to include these features
- CMS XML geometry files have improperly defined shapes and undefined object references
  - Fixes made or in progress
- Old, obscure code is difficult to migrate and test
Good Practices for Migration

- Perform **evaluation and test migration** to ensure toolkit will meet requirements
- Identify **special exceptions** in legacy code that will take most time to migrate
  - Assess whether special features can be dropped
  - If not, schedule sufficient resources for their migration
- Provide developers with migration examples and **instructions** to facilitate migration process
- Engage with toolkit developers to **enhance toolkit**
- Use migration as opportunity for overall software improvement
Acknowledgments

• Many thanks to the following for their contributions to this migration project:
  ▶ Sunanda Banerjee, FNAL
  ▶ Markus Frank, CERN, and the DD4hep team
  ▶ Andres Vargas Hernandez, Catholic University of America
  ▶ Vladimir Ivantchenko, CERN
  ▶ Sergio Lo Meo, INFN & ENEA
  ▶ Mircho Rodosov, Bulgarian Academy of Sciences
Summary

- DD4hep is a powerful toolkit for detector geometry
  - Fully featured
  - Committed to years of support and further innovation
  - Development team very responsive to user needs
- CMS adoption of DD4hep is a success story for community-supported software
  - DD4hep supports highly complex geometry of CMS detector
  - Migration process improved both CMS software and DD4hep itself
- Community-supported toolkits provide major benefits to HEP
Backup
Migration Challenges (1)

- TGeo used by DD4hep not thread-safe
  - ROOT fix made it thread-safe
- DD4hep required enhancement to become compatible with Geant 10.4
- Special shapes needed by CMS (cut tube, pseudo-trapezoid, and truncated tube)
  - DD4hep team added these shapes
- Incorrect polycone shapes in XML files have to be fixed
- Old, obscure code difficult to migrate and test
Migration Challenges (2)

- Reflection rotations used by CMS for sub-detectors with two mirror-image sides (left-handed coordinate system)
  - DD4hep implemented reflection rotations
- CMS uses both Geant4 and ROOT unit conventions (mm = 1 vs. cm = 1)
  - DD4hep enhanced to allow selection of units convention
- CMS DD allows reference to undefined geometric objects in XML
  - DD4hep requires all objects be defined before being referenced
  - CMS code required enhancement to safely process legacy XML files with undefined object references