The short story of EIC Central Detector integration software suite

(aka EIC Toy Model)

Alexander Kiselev

EIC@IP6 Tracking WG Meeting   May 12, 2021

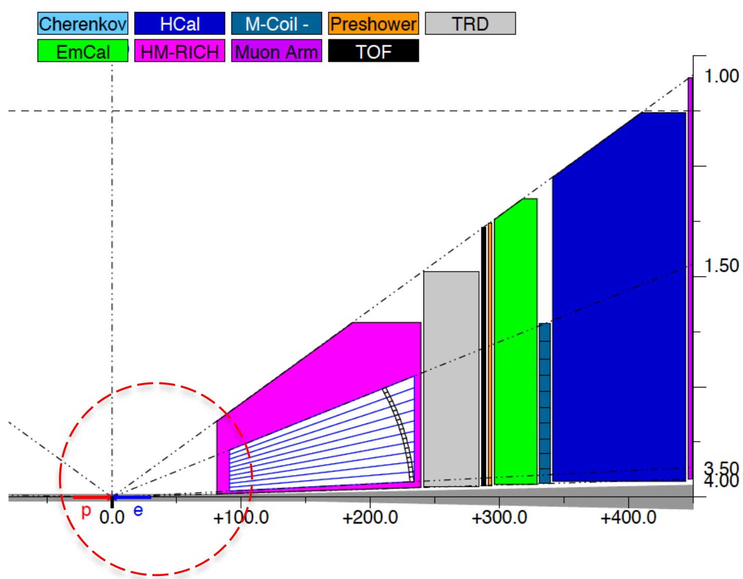
~May 2020: the starting point

Case #1: EIC greenfield solenoid specs document (detector composition)

	"Status"	Minimal	Default	"Ideal"
Muon Detector	optional	0	5 cm	
Hadronic calorimeter	mandatory	105 cm	105 cm	~150 cm
Correction coils	optional	0	10 cm	0 ☺
e/m calorimeter	mandatory	35 cm	35 cm	>35 cm
Preshower	optional	0	5 cm	
Time of flight	optional	0	5 cm	
GEM-TRD	TRD functionality is optional	~15 cm	45 cm	~60 cm
High-momentum RICH	mandatory	~120 cm	165 cm	

Table 1 Forward endcap space allocation.

Given the obvious space limitations in the forward endcap it seems to be reasonable to push the calorimetry equipment towards the very end of the +4.5 m zone from the start.

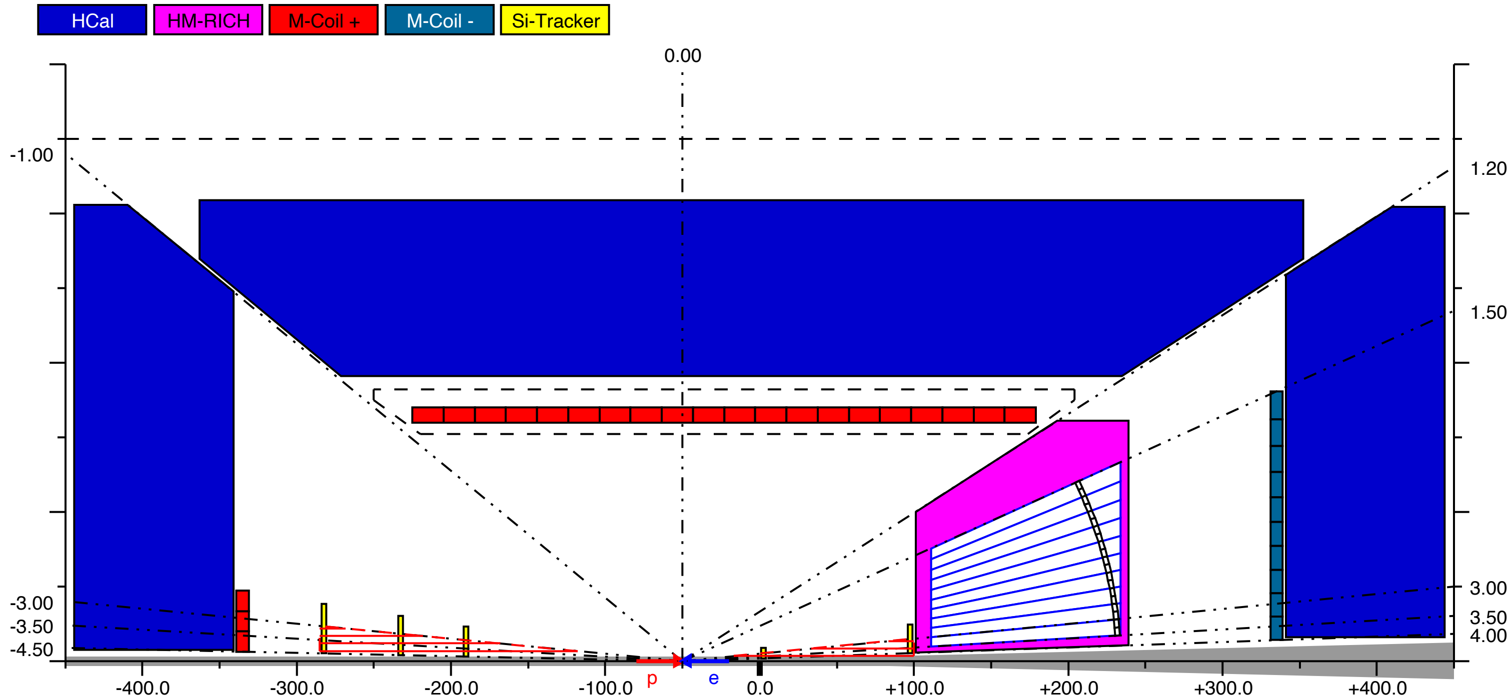


The detector composition, which includes all of the above-mentioned subsystems, with their respective **default** space allocations, requires ~3.75 m along the beam line direction, and has no real contingency included. In a "symmetric" configuration, shown in Figure 5, when the nominal IP is located in the center of the +/- 4.5 m region, provided by the accelerator layout, this would

A visual representation of the available space problem was required

~May 2020: the starting point

The primary goal: provide a set of cartoons like this



- **Just be able to illustrate several key features of the detector layout:**
  - A definitive location of the flux return elements (hadronic calorimeters)
  - A supposed location of the gaseous RICH (projective field required)
  - An optimal location of the nominal IP (split space between two endcaps)
  - The range where a constant magnetic field is desirable

An attempt to connect some of the other dots

Case #2: EIC Yellow Report Central Detector integration WG

Escalate & fun4all; migration process	Physics simulations & engineering design
Tracker, PID & Calorimetry detectors in GEANT	Ideal detectors & services / support
1-st & 2-d IR	$\|\eta\|<4.5$ & reality
EIC detector & greenfield solenoid design	Space available for the detectors & IR vacuum chamber

- One can easily identify several places with a lack of synchronization
- Some of them could seemingly be addressed in software, in a consistent way

Hack something more functional together *on top of the existing cartoon tool*?

EIC Toy Model: overview

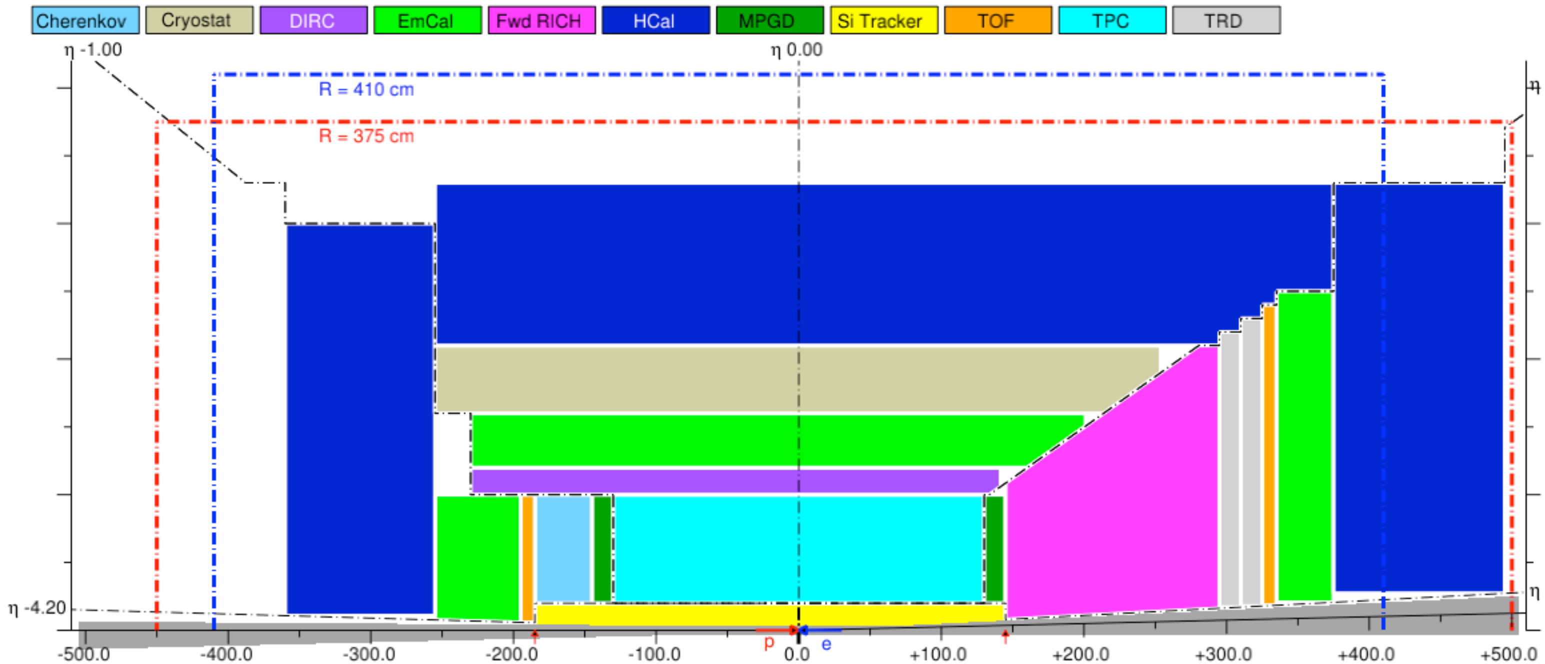
- **A tool to model & generate EIC Central Detector "templates" in a way:**
  - the new geometries (models) can be generated "quickly" …
  - … *by everybody*, and represented instantly in a WYSIWYG fashion
  - the sub-detector "container objects" are guaranteed to not overlap either with each other or with the IR vacuum chamber elements
  - technically they can be imported in GEANT frameworks in a consistent way and used as wrappers to the "real" sub-detectors
  - they can be exported in a CAD format to be used in the engineering design of the detector support structures and / or laying out services

- **Repository:** https://github.com/eic/EicToyModel
  - a README.md file ☺
  - example ROOT scripts
  - a standalone GEANT example, fun4all & g4e interfaces; containers, etc.
  - detailed API description

~3 months of active development, from scratch

The workflow

Create a model → **Save it as a .root file** → **Import in GEANT**



Note: simplified graphics (exportable objects only; no fine detail)

- https://github.com/eic/EicToyModel/blob/master/scripts/EIC-IR1-XX-v01d.C
- Minimal overhead to create a 2D scheme like this (ROOT scripting)
- Models could be saved, distributed and re-imported as .root files
- GEANT application: import .root file and **create volumes on the fly**

What was under the hood

- **A small ROOT-based C++ library, with several interfaces:**
  - GEANT4: dynamic conversion of a 2D cartoon into G4 “container volumes”
  - OpenCascade: export to STEP format

  - VGM: IR vacuum chamber TGeo -> G4 conversion for a “boolean cut”
  - VGM: direct import of EicRoot-like models into e.g. fun4all and “pure GEANT”

  - BeastMagneticField: ASCII field map import *(forward compatible format)*

- **Custom simplified IR vacuum chamber implementation (March 2020 model)**
  - II was parametric, so could be used to create e.g. a 50mrad IR layout

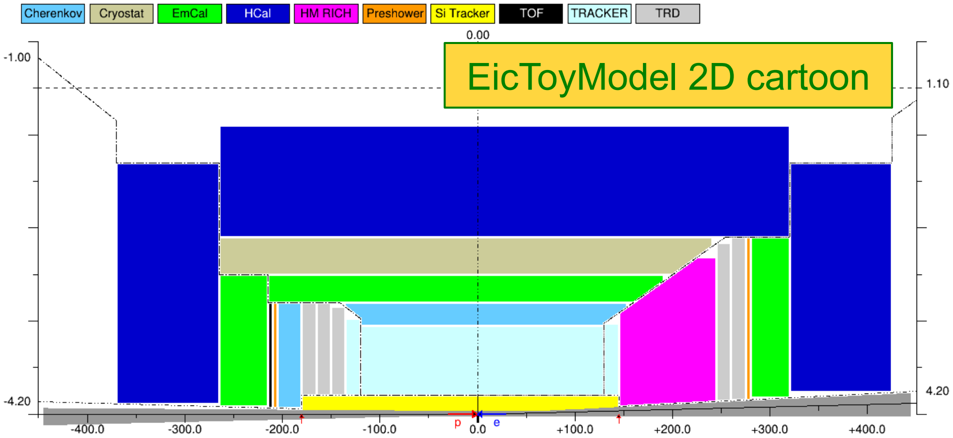
- **Limited set of interactive commands (IP shift, η range change, …)**

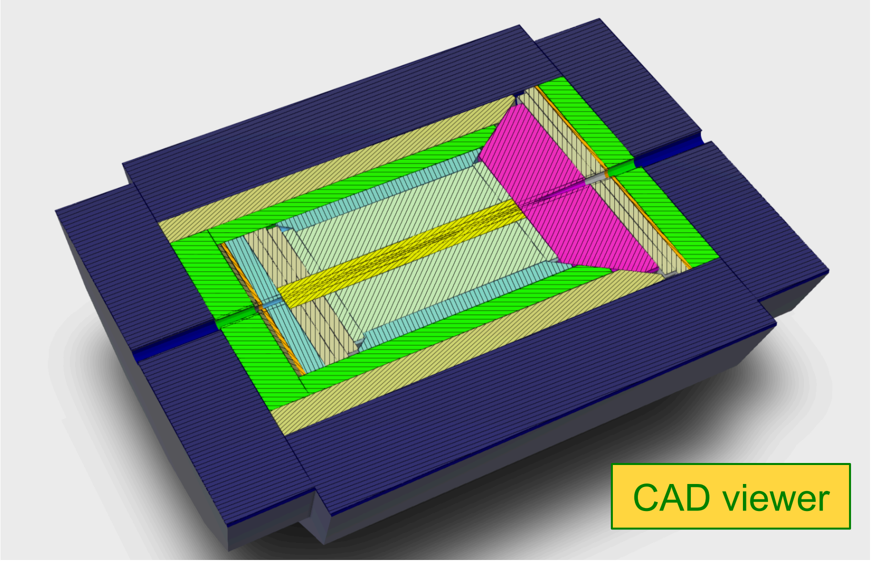
Step by step “how to run” instructions for various Linux installations

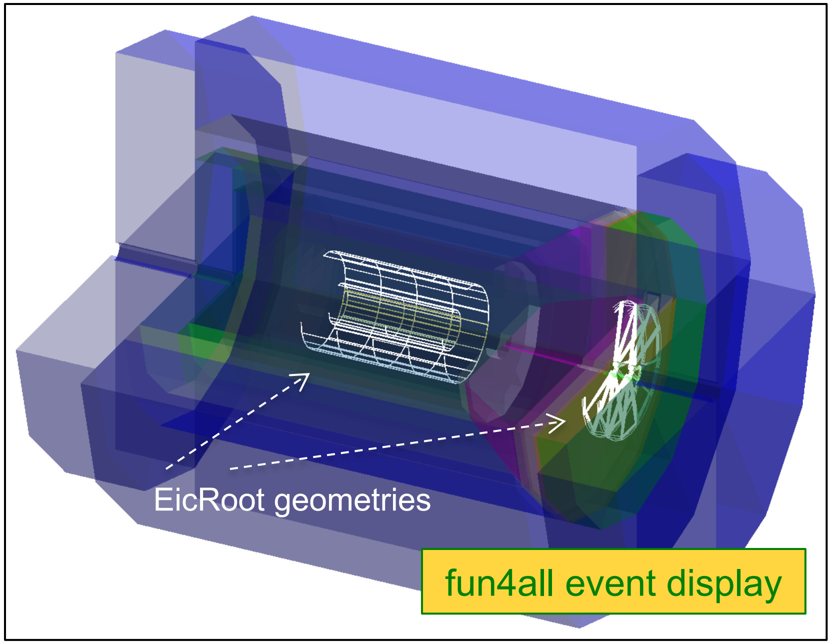
Note: at the end only a small fraction of the available functionality was ever used

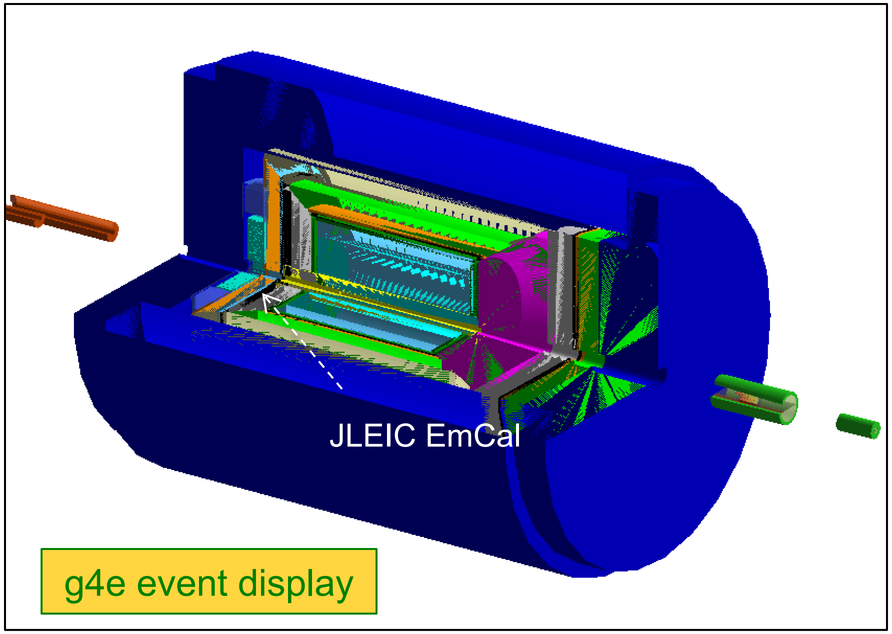
EIC Central Detector partitioning

- The same model in all cases









Coding overhead in GEANT

Excerpt from a modified working calorimetry code:

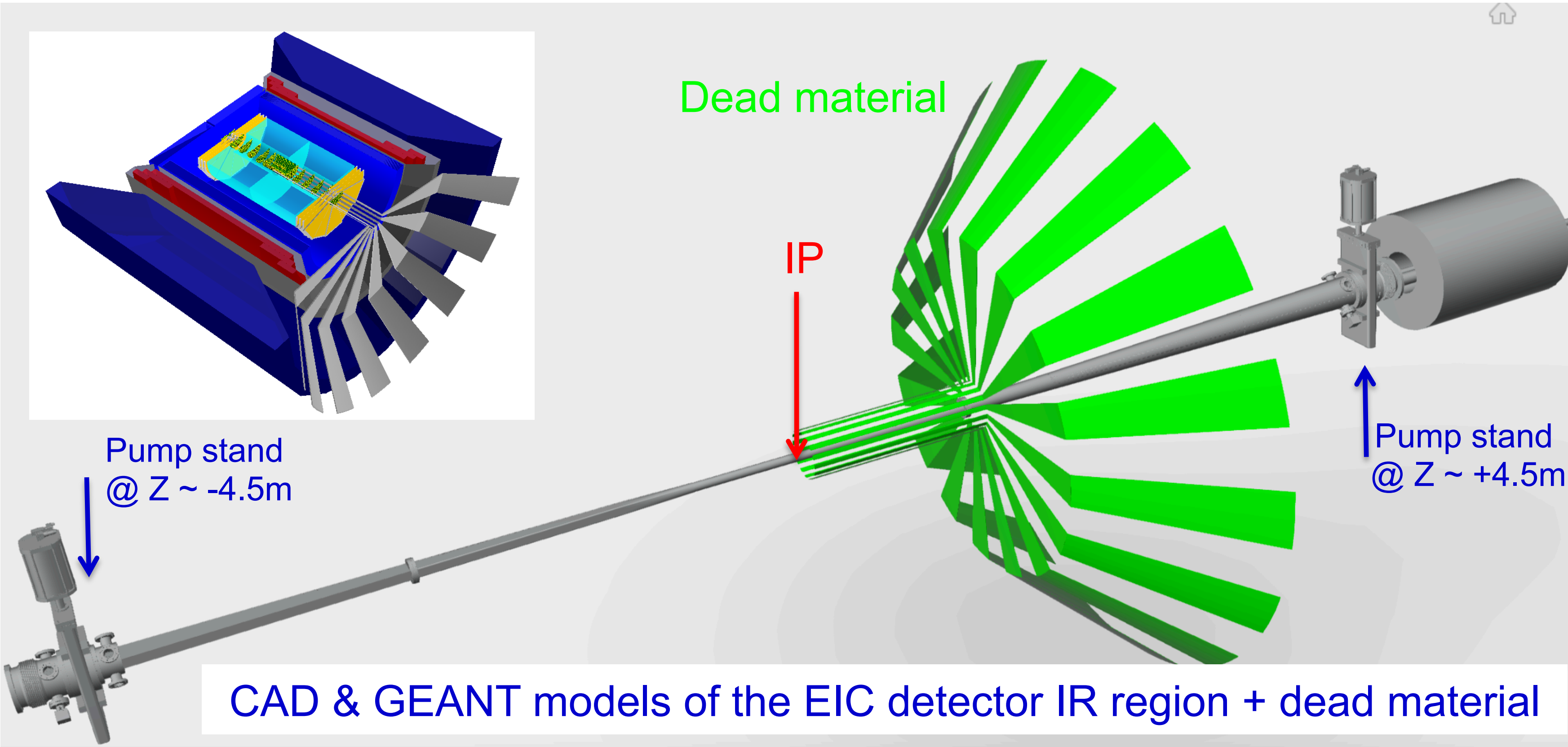
```
// Construct the integration volumes geometry, internally;
TFile fin(argv[1]);
dynamic_cast<EicToyModel *>(fin.Get("EicToyModel"));
eic->Construct();
// Populate G4 world by these volumes;
eic->PlaceG4Volumes(expHall_phys);

// Place "MyHCal" tower matrix into the integration volume bubble instead of the world;
new G4PVPlacement(0, G4ThreeVector(0, 0, zOffset), myhcal_log, "MyHCal", expHall_log,    false, 0);
auto hcal_bubble_log = eic->fwd()->get("HCal")->GetG4Volume()->GetLogicalVolume();
new G4PVPlacement(0, G4ThreeVector(0, 0,       0), myhcal_log, "MyHCal", hcal_bubble_log, false, 0);
```

This part should be taken care of by the framework

- Immediate migration was not mandatory for everybody
  - Integration bubbles can be imported into a framework one by one
- Bubble size (and location) could be polled *(feature was never requested)*
  - Parametric detectors can be implemented in a proper way
- If the community preferred to use GDML files instead, was also possible

Support, services, detector frames?



- Support structure:
  - Generic part (outside of the integration volumes): engineering effort
  - Matching detector-specific part (inside the integration volumes ?)
- Services: should be configurable, accumulating from / to “inner” detectors
- Detector frames: should naturally come together with the active volumes

Why did it fail in the YR process?

- Was rolled out end of July 2020 in a pretty much useable state
  - Still too late for the YR integration / simulation effort?
- Was easy to use and had very little migration overhead
  - Still an overhead; also, define “easy” and “little”
- Was a very thin software layer, seemingly able to serve as a “grand unification scheme” for the YR simulation efforts
  - Yeah, but there was no strong motivation to unify them in the YR process, right?
- The developer and the Integration WG convener was the same person (AK)
- The suite had a git repo, very comprehensive documentation, step by step examples including “you only need five minutes to get this cartoon” ones
- Was presented at several YR WG and tri-monthly meetings
  - Nope, this does not help

Take this as a lesson, and think why and where May 2021 is any different from May 2020

EicToyModel use in the EIC@IP6 work

- Integration volumes?
  - Not much to unify at this stage -> just use dd4hep (or any other) description

- Interactive detector composer, IR material scan, export to CAD -> nope!
  - Other tools are promoted strongly

- EicRoot configurable tracker models in fun4all
  - ITS2 model for sure can be used
  - GEM wedges perhaps; barrel MM is implemented elsewhere

It looks like everything is available in fun4all singularity environment; needs to be verified

Creating ITS2-like objects in fun4all

- *This code caption is part of the fun4all simulation script*

```
{
  VstGeoParData vst;
  vst.SetGeometryType(EicGeoParData::SimpleStructure);

  auto ibcell = new MapsMimosaAssembly();

  //vst.AddBarrelLayer(ibcell, 6*12, 14, 6*23.4 * etm::mm, 14.0, 0.0);
  vst.AddBarrelLayer(ibcell, 4*20, 14, 4*39.3 * etm::mm, 14.0, 0.0);

  auto mid = eic->mid()->get("TRACKER")->GetG4Volume();
  vst.PlaceG4Volume(mid, true, 0, new G4ThreeVector(0, 0, 0));
}
```