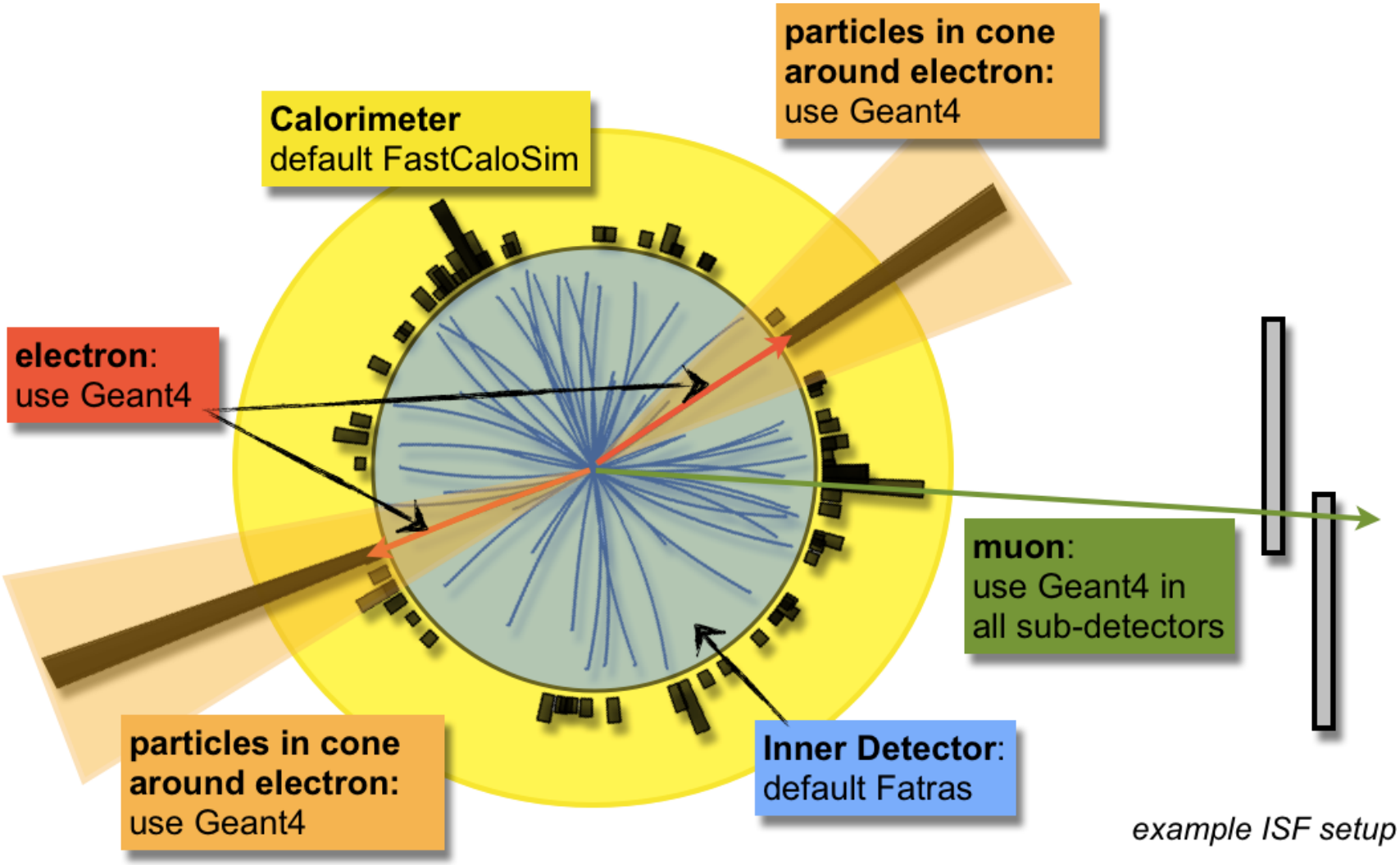
Simulation Infrastructure

Software Technical Interchange Meeting
9-14 November 2015 (LBNL)

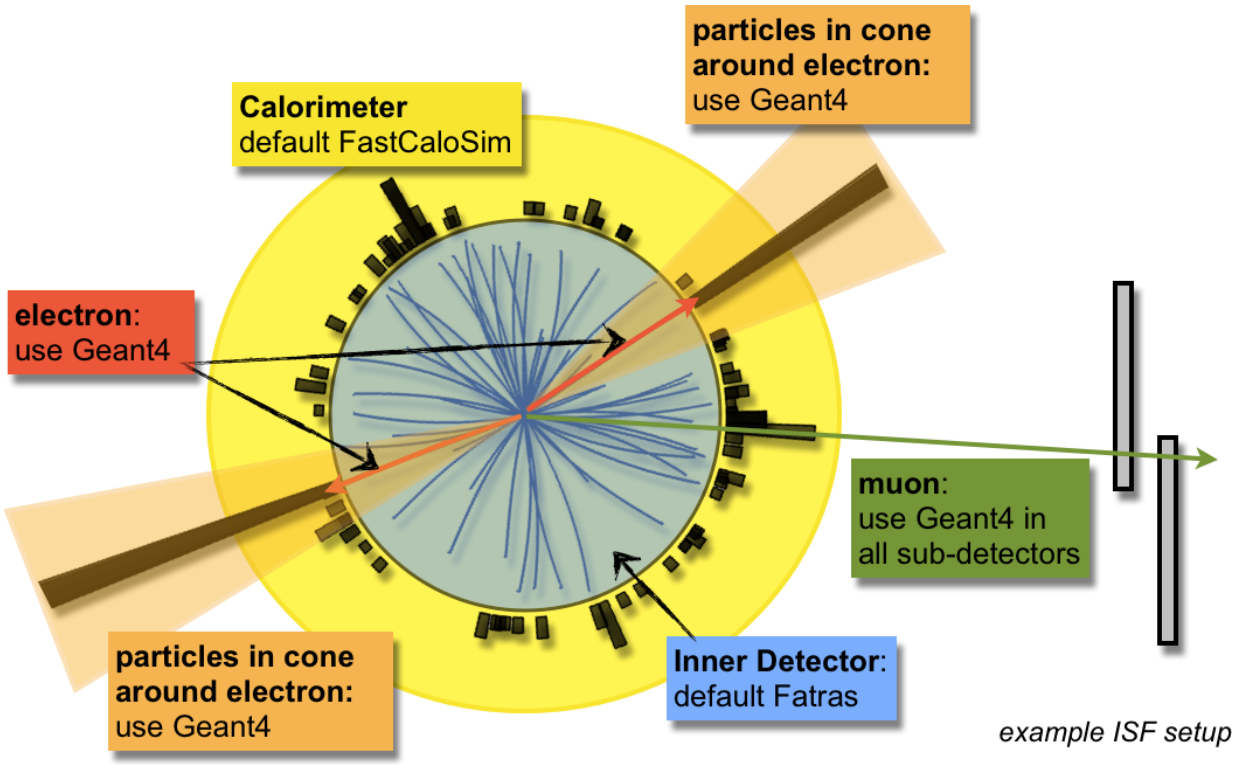
Elmar Ritsch (CERN)
for the Simulation Team

Integrated Simulation Framework (ISF)



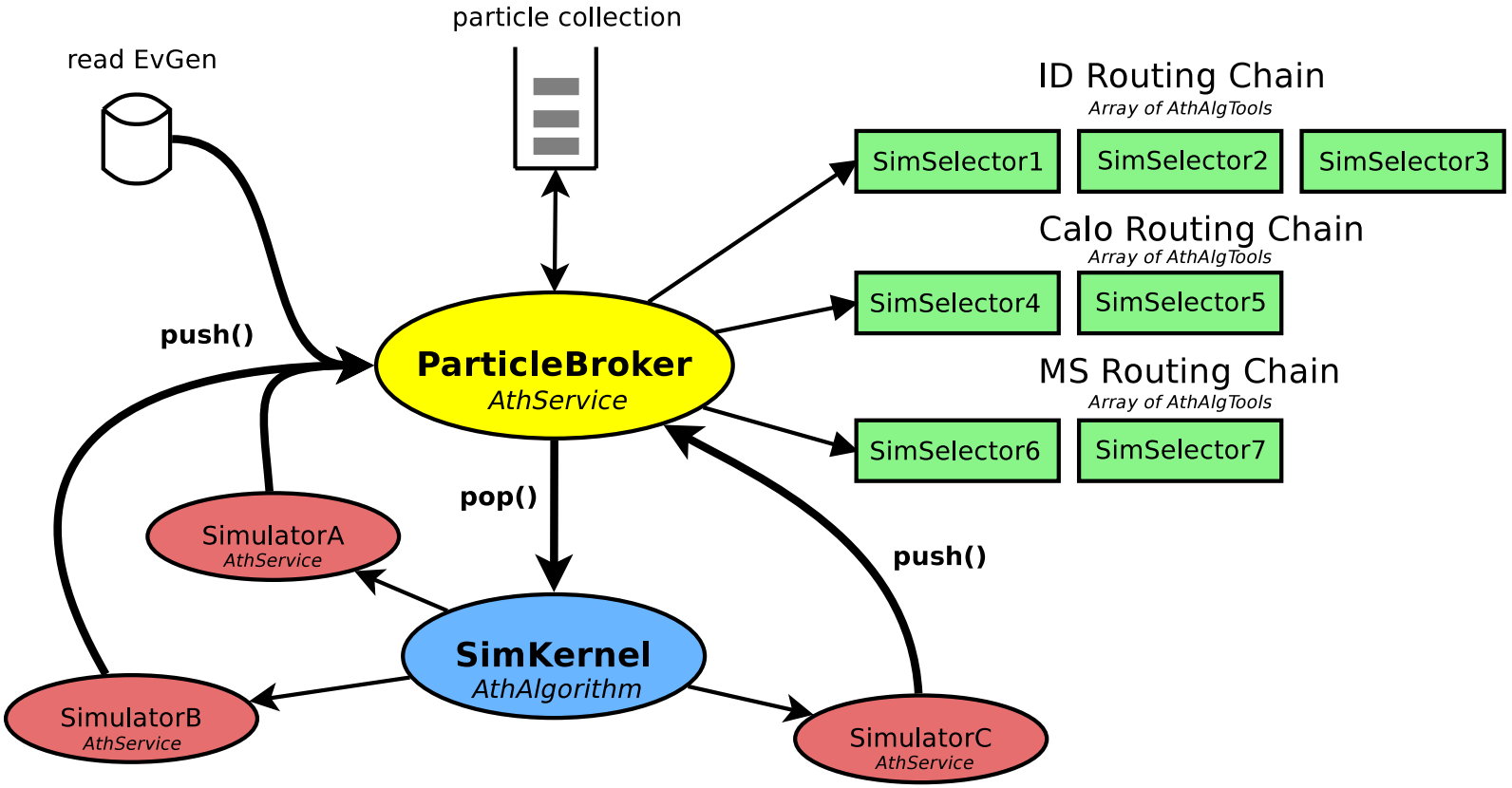
example ISF setup

Integrated Simulation Framework (ISF)



- ISF is **default framework** used for simulation production: full and fast simulation
- Possible to define **regions-of-interest**

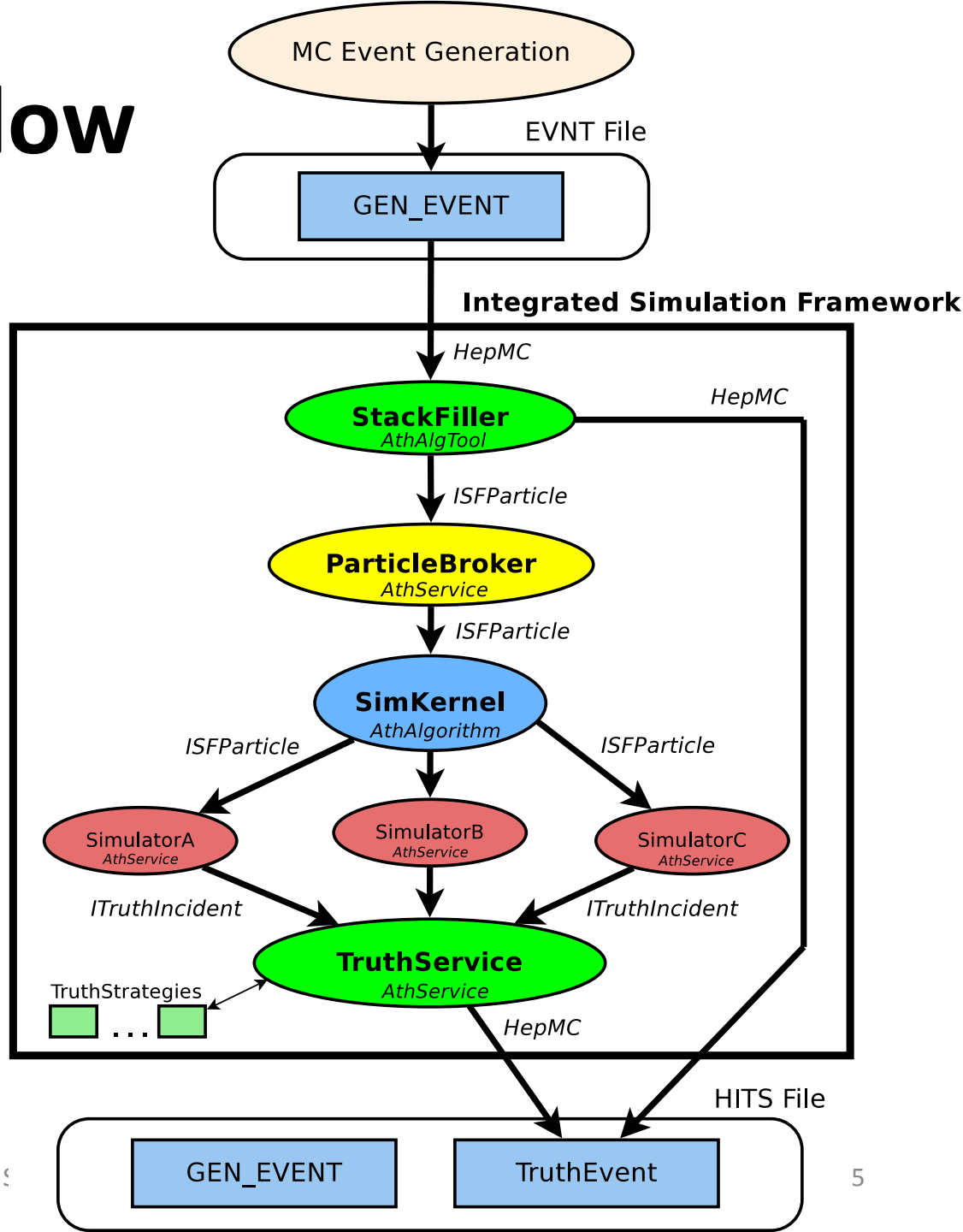
ISF Structure



- SimKernel is **sole** AthAlgorithm
- SimKernel implements **particle loop** and sends particles to respective simulators (AthServices)

ISF MC Truth Flow

- **One common TruthService** used by all simulators
- Enables **consistent application of TruthStrategies** across different simulators
- **TruthService responsible for building consistent MC truth tree** with individual particles from different simulators



Geant4 Integration into ISF

- **Wrapped existing FADS-based Geant4 setup** and configuration into ISF-compatible AthService
  - FADS being replaced now (more later in this talk)
- An Athena event may be **split up into multiple G4Events**
  - This **might allow sub-event parallelism**, but we haven't worked out all the issues yet..
    - E.g. many threads writing to same containers, or merge step?
  - NB: when running FullG4 simulation in ISF:
    1 Athena event corresponds to 1 G4Event
    - Reduce overhead (a few percent) and to generate bit-wise identical output with pre-ISF Geant4 (for initial validation)

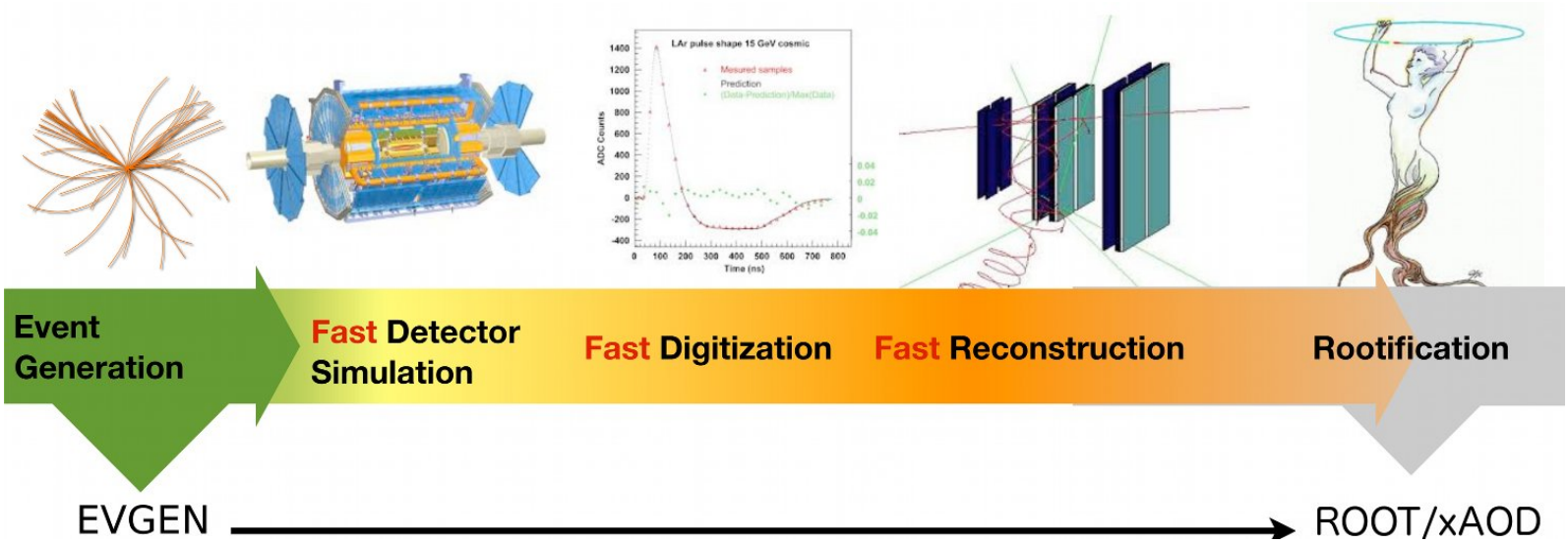
FastCaloSim Integration into ISF

- **FastCaloSim** = parameterized ATLAS calorimeter simulation
  - Geant4 (ID + μ) + FastCaloSim = **ATLFASTII**
- **Wrapped existing FastCaloSim** implementation and configuration into ISF-compatible AthService
  - Re-implementation and re-parametrization of FastCaloSim is currently ongoing effort
- Setup of ATLFASTII much **more straight-forward due to ISF**
  - Previously two separate simulation jobs were needed to run AFII

Fatras Integration into ISF

- **Fatras** = fast ATLAS Tracker Simulation
  - Uses ATLAS tracking extrapolator and geometry
  - Fatras (ID + μ) + FastCaloSim = **ATLFASTIIF**
- **Re-implemented Fatras from scratch**
  - Provides native ISF integration
- Fatras is currently **actively being worked on** to get ready for production use
- Fatras is **heavily used by upgrade community** because implementation of geometries much simpler than full-detail version for G4

The FastChain



- Goal: from EVGEN to xAOD in **one step**
- Combining **fast simulation** with **fast digitization** and **fast reconstruction**
- **Flexibility of ISF** made setting up simulation easy

The FastChain Setup

- On-the-fly **pileup generation** with Pythia 8
  - Apart from hard-scatter EVGEN dataset **no additional input files required**
- Hard-scatter and pileup being treated differently:

Hard-scatter Particles	Pileup Particles
fast simulation	fast simulation
standard digitization	fast digitization
standard reconstruction	fast reconstruction

- Very first **timing measurement**:
  ~30 sec/event for EVGEN→AOD with μ=40
  - Only standard reconstruction used
  - ~11 sec/event for simulation and ~18 sec/event for standard reconstruction

Framework for ATLAS Detector Simulation (FADS)

- Current default framework which **integrates Geant4 into ATLAS code**
- Implemented ~2003
- FADS uses **Python-driven C++ classes to setup G4**
- FADS **initializes G4 partially steered by the Python layer and partially steered by C++** during the Athena "initialize" step
  - Even though G4 mandates a specific order in which things are to be initialized, FADS makes it awkward/difficult to understand in which order this initialization is carried out by the ATLAS code
- FADS relies on **instantiating static objects when libraries are loaded**
  - Had cases where loading of a library caused the behaviour of G4 to change (G4MultiNavigator got inadvertently activated)

→ **Migrating away from FADS to a cleaner and more modern ATLAS Geant4 framework**

Z. Marshall
S. Farrell
A. Dell'Acqua
A. Di Simone
E. Farina
JC et al

FADS Replacement

- The new G4Atlas Framework (fancy name TBD) is a **replacement for FADS**, but not the ISF.
- Aim to **take advantage of the current features of Athena/GAUDI** and share code with the ISF where possible.
- **Replacing Python-driven C++ classes by pure C++** implementation for initialization
  - Individual tools are now taking care of different parts of the G4 initialization
  - Order of tool retrieval steered through one central service
- Designing the new framework with an **eye on the future**.
  - Built-in thread safety from the start.
  - Design with multi-threading in mind generally.
- **Migrating each part of FADS in turn and validating the simulation** output at each step.
- Continuously **reviewing design**.

slide by J. Chapman

Extra Slides

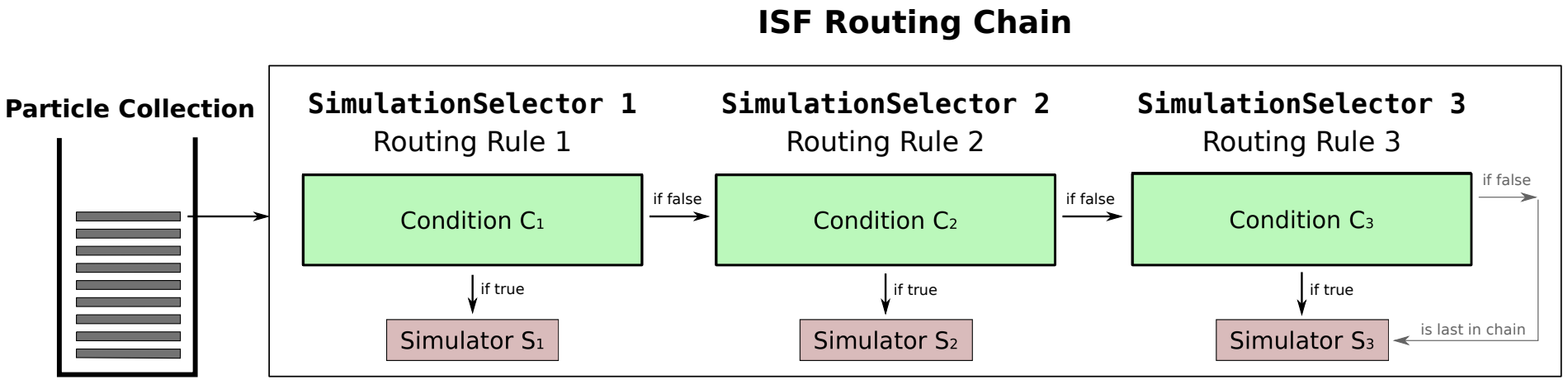
Sensitive Detector Re-write

- **Static sensitive detector classes from FADS have been replaced by Sensitive Detector AlgTools**, which own G4VSensitiveDetector implementations (one per GAUDI-thread).
- Another issue we had **in the past was not being able to assign multiple SDs to a given G4LogicalVolume**.
  - The use-case here was the ability to write out more detailed 'calibration hits' as well as standard hits from the calorimeter.
- **Discussion with the G4 experts resulted in the G4MultiSensitiveDetector class** (officially available in G4 10.2 onwards) which can be associated with a volume and owns other sensitive detector instances.
- **SG::WriteHandles are used get around issues with G4 and Athena event loops not necessarily being aligned** and ensure that the output writing is thread-safe.

PhysicsList Config Re-write

- **FADS Approach:**
  - **FadsPhysicsList objects** (objects that are small that own the actual G4VPhysicsList implementation, and build it on demand) **are created as static objects** that get built when the library is loaded. They **register themselves in a catalogue.**
  - **At runtime we pick a list from the catalogue** and ask it to build the G4VPhysicsList implementation.
- **New Approach:**
  - **Wrap the G4PhysListFactory in an AtlasPhysListFactory** which uses the G4PhysListFactory to build official physics lists and variants and knows how to build any ATLAS-defined physics lists.

ISF Routing Chain



- One routing chain per sub-detector volume