Key4hep

The common software stack for future experiments

Thomas Madlener

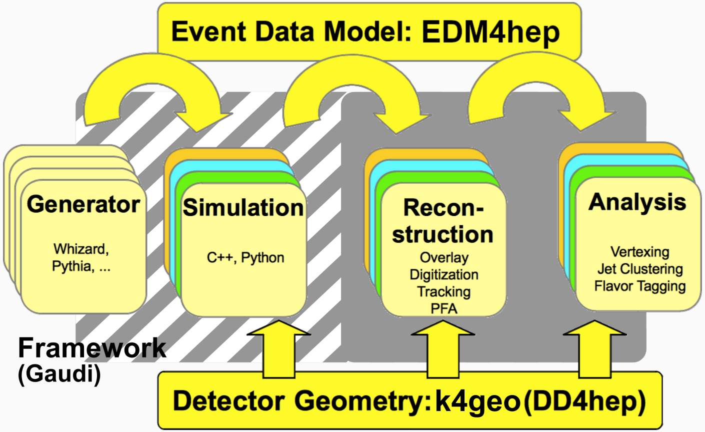
for the Key4hep developers

WLCG/HSF Workshop

May 14, 2025

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 101004761.

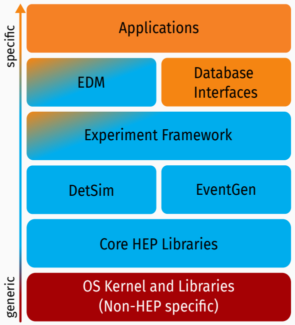
From generation to analysis - the general workflow



- Many steps involved from generating events to analyzing them
- Hundreds of SW packages
    - Building & deploying
    - Consistency
    - Reproducibility
- Try to give an overview of the **Key4hep** SW ecosystem

Key4hep - A (very) brief introduction

- Future detector studies rely on well maintained software for studying their potential
- Maintenance of a consistent HEP SW stack is non-trivial
    - Ecosystem of interacting components
- Sharing the burden allows everybody to reap the benefits
    - Make best use of scarce (human) resources
- **Regular contributions from ILC, CLIC, FCC, CEPC, EIC, (MuonCollider), ...**
- Support from major R&D initatives
    - CERN R&D for Future Experiments, AIDAinnova WP12, ECFA



Key4hep goals

- Provide and maintain a consistent SW stack that allows to do physics studies for **all projects**
- Ensure interoperability of the necessary building blocks
- Reuse existing solutions where possible
  - A lot of experience from LHC experiments and LC communities
- Focus new developments on EW/Higgs factory specifics
- Share knowledge, processes, workflows and resources
  - Best practices, tutorials, documentation, …

Non-goal

- Develop and maintain project specific software and workflows

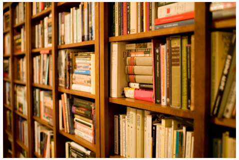
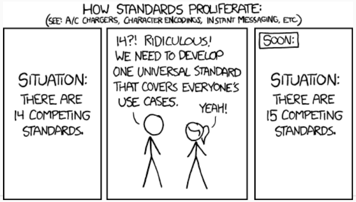
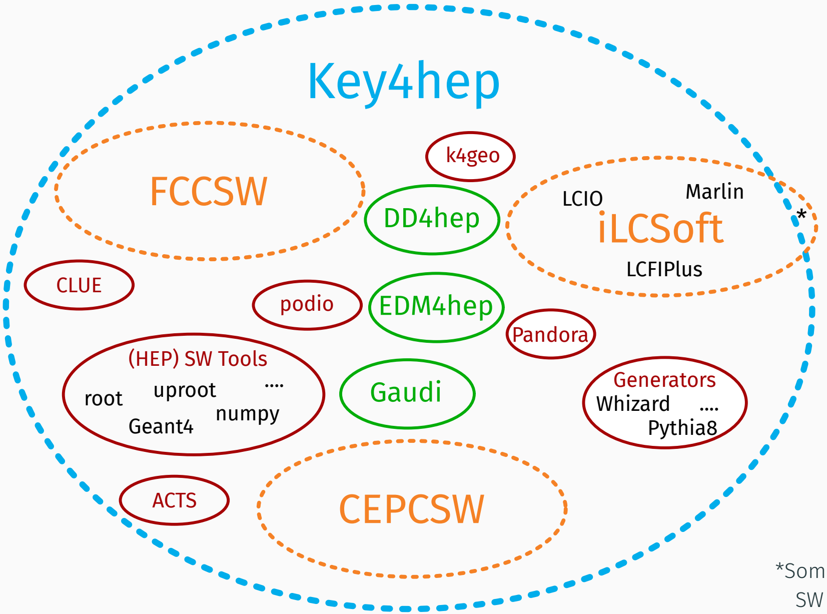


Photo by Stewart B. / CC-BY


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.
xkcd.com/927

Key4hep (simplified) overview

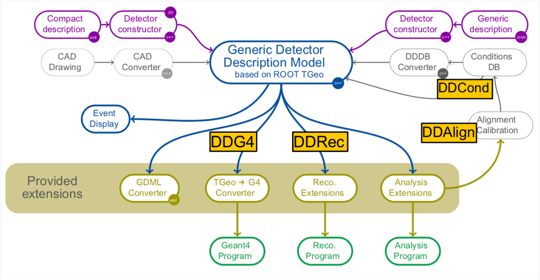


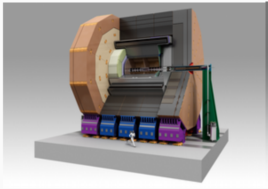
*Some testbeam related SW not yet included

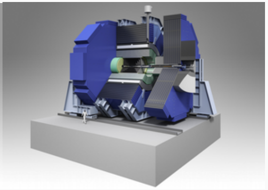
DD4hep - Detector description

dd4hep.web.cern.ch

- Complete detector description
  - Geometry, materials, visualization, readout, alignment, calibration, …
- From a **single source of information**
  - Simulation, reconstruction, analysis
- Comes with a powerful plug-in mechanism that allows customization
- More or less “industry standard” now
  - FCC, ILC, CLIC, EIC, LHCb, CMS, ODD, …
- `ddsim` - standalone simulation executable

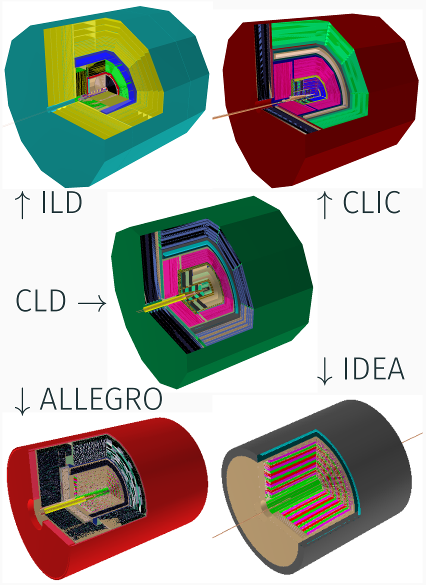




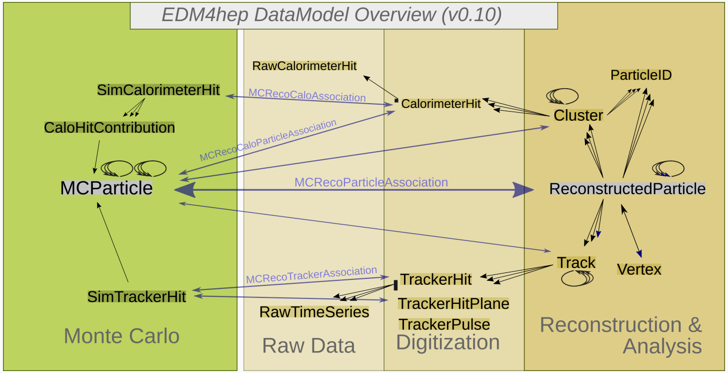


k4geo - The detector geometry repository

- `iLCSoft/lcgeo` $\rightarrow$ `key4hep/k4geo`
- Many existing detector models from LC studies
- Migration of detector concepts from [HEP-FCC/FCCDetectors]
    - ALLEGRO (Noble liquid)
    - IDEA
- New ARC detector concept in CLD
- Central repository for detector models (and drivers)



EDM4hep - The common EDM for Key4hep



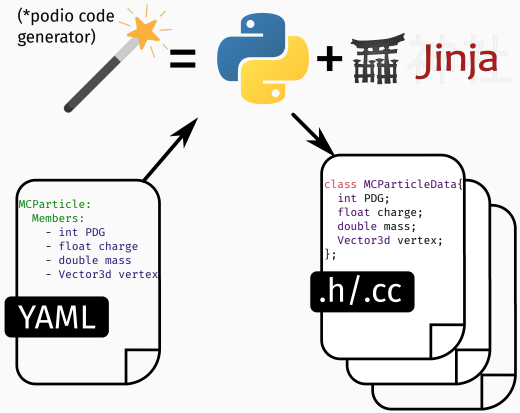
key4hep/EDM4hep

edm4hep.web.cern.ch

- Based on `LCIO` and `FCC-edm`
  - Focus on usability in analysis
- Quite stable over the last two years
  - **Some breaking changes recently for v1.0!**
- Can easily be extended
  - Used by EDM4eic
  - Prototyping!
- Generated via `podio`

The podio EDM toolkit

- Implementing a performant event data model (EDM) is non-trivial
- Use `podio` to generate code starting from a high level description
- Provide an easy to use interface to the users
- Main customers and feature drivers
  - key4hep/EDM4hep
  - eic/EDM4eic



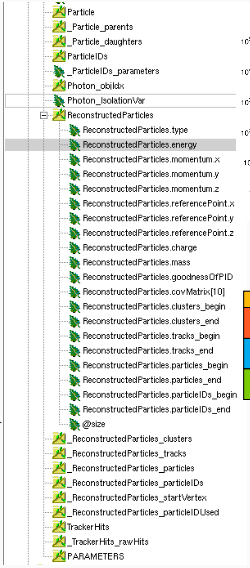
```
MCParticle:
  Members:
    - int PDG
    - float charge
    - double mass
    - Vector3d vertex
```

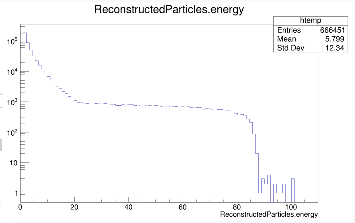
```
class MCParticleData{
  int PDG;
  float charge;
  double mass;
  Vector3d vertex;
};
```

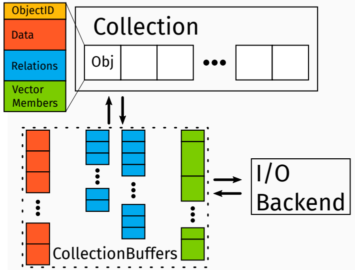
AIDASoft/podio

podio supports different I/O backends

- Default **ROOT** backend
  - Effectively flat `TTree` / `RNTuple`
  - Files can be interpreted **without EDM library**(!)
  - Can be used in `RDataFrame` (**FCCAnalyses**) or with `uproot`
  - Also with Julia
- Adding more I/O backends is possible
  - Alternative SIO backend exists
  - Working on `RDataSource` for better RDataFrame integration
- Generated interfaces provide many “convenience features”


```
Particle
_Particle_parents
_Particle_daughters
ParticleIDs
_ParticleIDs_parameters
Photon_objIdx
Photon_IsolationVar
ReconstructedParticles
    ReconstructedParticles.type
    ReconstructedParticles.energy
    ReconstructedParticles.momentum.x
    ReconstructedParticles.momentum.y
    ReconstructedParticles.momentum.z
    ReconstructedParticles.referencePoint.x
    ReconstructedParticles.referencePoint.y
    ReconstructedParticles.referencePoint.z
    ReconstructedParticles.charge
    ReconstructedParticles.mass
    ReconstructedParticles.goodnessOfPID
    ReconstructedParticles.covMatrix[10]
    ReconstructedParticles.clusters_begin
    ReconstructedParticles.clusters_end
    ReconstructedParticles.tracks_begin
    ReconstructedParticles.tracks_end
    ReconstructedParticles.particles_begin
    ReconstructedParticles.particles_end
    ReconstructedParticles.particleIDs_begin
    ReconstructedParticles.particleIDs_end
    @size
_ReconstructedParticles_clusters
_ReconstructedParticles_tracks
_ReconstructedParticles_particles
_ReconstructedParticles_particleIDs
_ReconstructedParticles_startVertex
_ReconstructedParticles_particleIDUsed
TrackerHits
_TrackerHits_rawHits
PARAMETERS
```



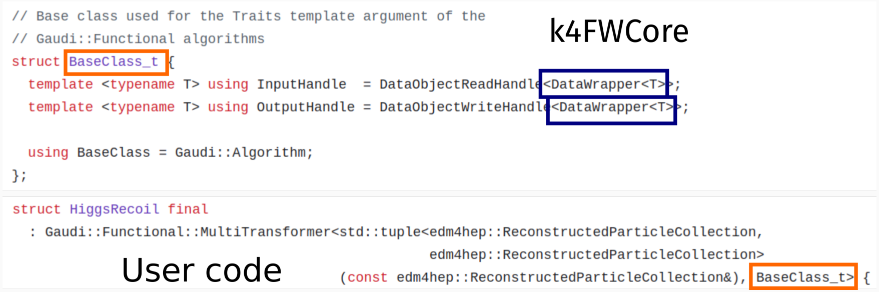


Experiment Framework

- `Gaudi`, originally developed by LHCb, now also used by ATLAS, FCCSW and smaller experiments
    - Supports concurrency
    - "Battle-proven" from data taking during LHC operations
- Key4hep has decided to adapt `Gaudi` as its experiment framework
    - Contribute to its development where necessary
- Integration and migration of iLCSoft algorithms into Key4hep with the help of a `Marlin`$\rightarrow$`Gaudi` wrapper

k4FWCore - The core Gaudi components

- Data Service for EDM4hep / podio inputs & outputs
  - Dedicated `DataHandles` to handle `podio::CollectionBase`
- `k4run` for running Gaudi options files
  - Some special casing for handling Gaudi exit codes
  - Custom arg parser to make algorithm parameters configurable
- Support for `Gaudi::Functional`
  - Ongoing work for proper multithreading support



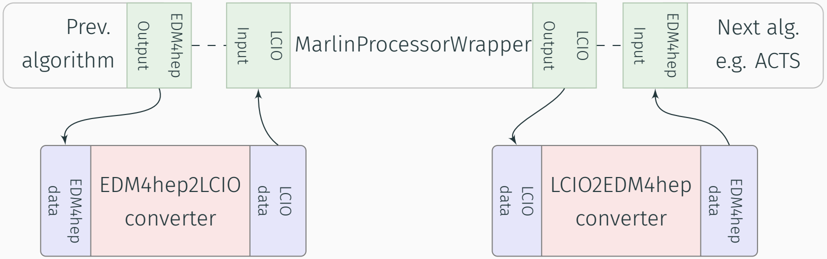
```cpp
// Base class used for the Traits template argument of the
// Gaudi::Functional algorithms
struct BaseClass_t {
  template <typename T> using InputHandle  = DataObjectReadHandle<DataWrapper<T>>;
  template <typename T> using OutputHandle = DataObjectWriteHandle<DataWrapper<T>>;

  using BaseClass = Gaudi::Algorithm;
};
```

```cpp
struct HiggsRecoil final
  : Gaudi::Functional::MultiTransformer<std::tuple<edm4hep::ReconstructedParticleCollection,
                                                   edm4hep::ReconstructedParticleCollection>
                                        (const edm4hep::ReconstructedParticleCollection&), BaseClass_t> {
```

k4MarlinWrapper

- Wraps **Marlin processor** in a Gaudi algorithm and allows to **run them unchanged**
- Automatic, on-the-fly conversion between LCIO and EDM4hep
    - (Incrementally) build parallel events in memory
- **Allows to "mix and match" existing reconstruction algorithms with new developments**
    - Working horse for many studies at the moment

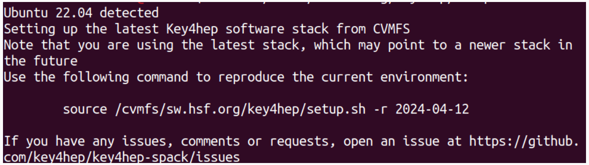


Keyhep releases and nightlies

- (Rolling) latest release of the complete Key4hep software stack
  - Full stacks for AlmaLinux9, CentOS7, Ubuntu22.04

```
/cvmfs/sw.hsf.org/key4hep/setup.sh
/cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh
```

- Documentation
  - key4hep.github.io/key4hep-doc
  - Includes tutorials & How-tos
  - Partially checked by CI



```
Ubuntu 22.04 detected
Setting up the latest Key4hep software stack from CVMFS
Note that you are using the latest stack, which may point to a newer stack in
the future
Use the following command to reproduce the current environment:

        source /cvmfs/sw.hsf.org/key4hep/setup.sh -r 2024-04-12

If you have any issues, comments or requests, open an issue at https://github.
com/key4hep/key4hep-spack/issues
```

- **Release early and release often**
  - Make fixes available early
  - Discover problems and collect feedback as early as possible

Spack for Key4hep

- Spack is a package manager
  - Independent of operating system
  - Builds all packages from source
- Originaly developed by the HPC community
  - Emphasis on dealing with **multiple configurations** of the same package
- Basic building block is a formalized build procedure $\rightarrow$ **spack recipe**
  - Build instructions, dependencies, versions and location of source code
  - $\sim$ 8000 packages currently available from spack
  - Many Key4hep packages in key4hep/key4hep-spack
- The whole Key4hep software stack can be built from scratch using spack

```
spack install key4hep-stack
```

Spack experiences

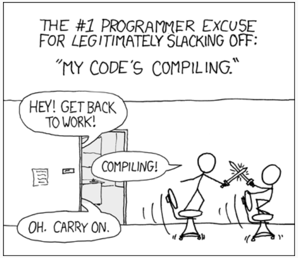
- Spack is still under active development
  - No split between package updates and package manager updates (yet)
  - **Resorted to picking Spack commit and cherry picking package updates**
  - Move spack for nightlies every few weeks
- New(-ish) spack *concretizer* works quite well
  - Re-uses existing (installed) packages where possible
  - Might silently drop some *variants* (feature flags) if not explicitly required
  - **Specific configuration for some packages**
- Adding a new compiler / OS build usually uncovers some issues
  - Recently managed to build (slightly reduced) Key4hep stack on macOS
- Working quite well for our purposes

Developing Key4hep

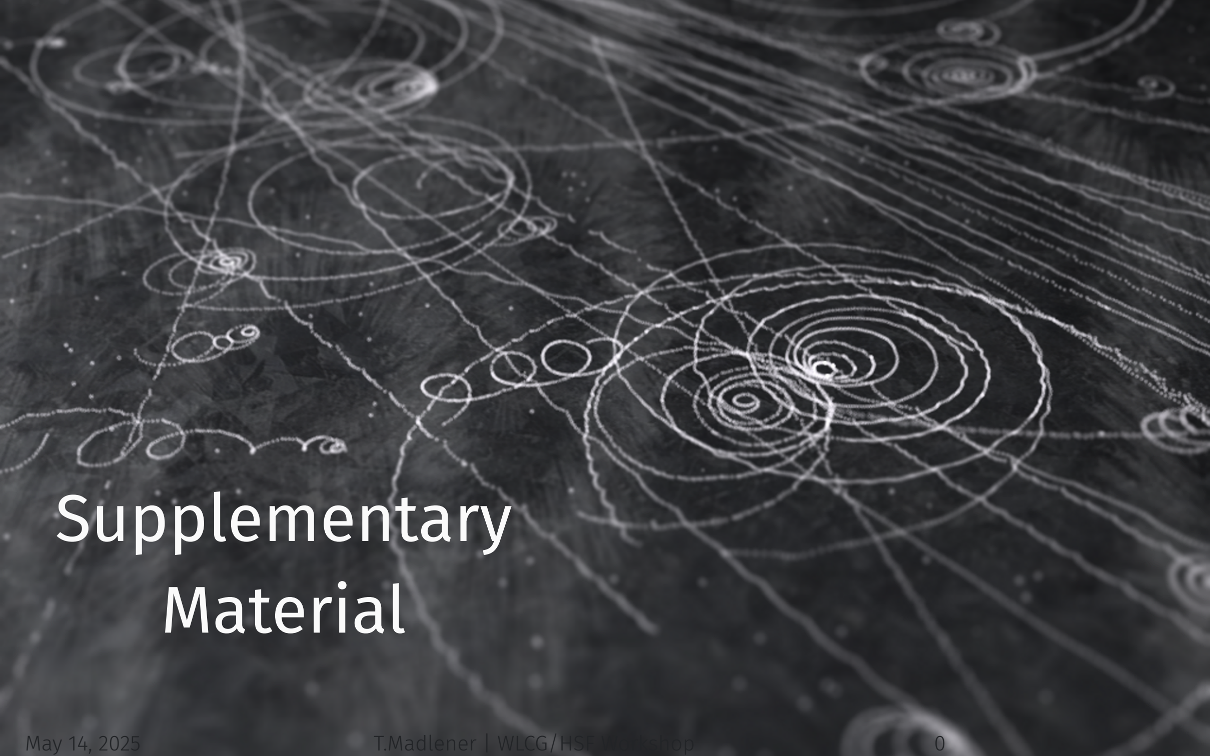
- For single packages / repositories `k4_local_repo` (bash function)
  - Changes environment variables to point to a local build / install
  - Works well for “leaf packages”
  - Dependency management for multiple packages offloaded to user
- Can also use *spack develop* environments
  - Not really the main intended way of using spack
  - Need to know quite a bit of spack for initial setup
  - Works quite well once setup
  - Used for features spanning multiple packages
- Works but there is room for improvement$^{\text{IMHO}}$
  - Could make the onboarding experience better

Summary

- Key4hep aims to provide a common software stack for all future collider projects
- **Very successful in bringing together communities and focusing on common approaches**
- In use for physics studies by several communities already
- Finalizing core components
- Ongoing efforts towards more "native Key4hep algorithms"
- No shortage of work



xkcd.com/303



Supplementary Material

Spack for nightlies (and releases)

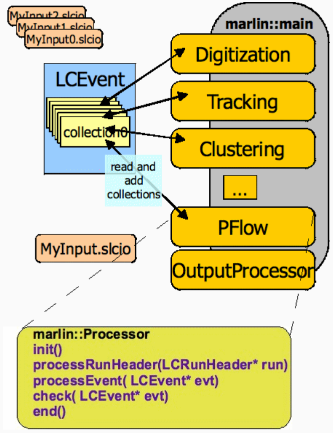
- Building everything from scratch every day is not feasible
- Spack does not separate package (version) updates from core updates
- Pick a spack commit / tag and build a “base stack” with major dependencies
  - Use all previosly installed packages as `spack upstream` in nightlies
  - Move this roughly monthly to newer spack version
- Cherry-pick package version updates / fixes for *central spack repository*
- Use latest version of Key4hep package repository
- Define latest commit as “version” for Key4hep packages
  - Simple script to collect them all

Key4hep packages

- `k4FWCore` key4hep/k4FWCore
  - Core Key4hep framework providing core functionality, e.g.
    - Data Service for EDM4hep / podio inputs
    - Overlay for backgrounds
- `k4SimDelphes` for Delphes fast simulation key4hep/k4SimDelphes
- `k4MarlinWrapper` Marlin proc. wrapper key4hep/k4MarlinWrapper
- Many packages migrated from FCCSW to Key4hep
  - `k4SimGeant4` for Geant4 simulation integration HEP-FCC/k4SimGeant4
  - `k4Gen` for generic generator interface HEP-FCC/k4Gen
  - ...
- Ongoing work to integrate more components
  - ACTS tracking framework acts-project/acts | key4hep/k4ActsTracking
  - CLUE fast clustering algorithms .cern.ch/kalos/CLUE | key4hep/k4CLUE

Reconstruction and Analysis with Marlin

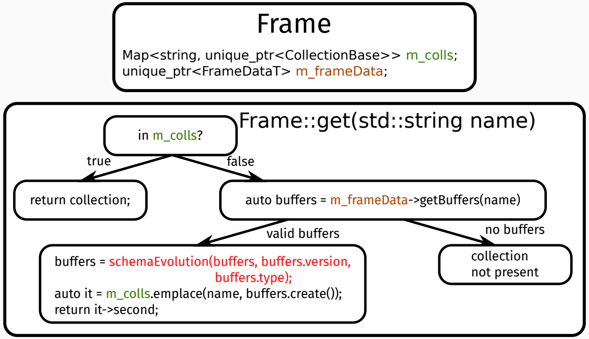
- `Marlin` framework from iLCSoft has been tried and tested in ILC and CLIC studies
  - Marlin *Processors* are the working units
- Complete (low level) reconstruction chain available in iLCSoft
  - Digitization, tracking, particle flow (Pandora), …
- Many high level analysis algorithms for various tasks
  - Jet flavor tagging, isolated lepton finding, …
- On a high level very similar to Gaudi framework
  - Differences emerge at various “lower” levels



```
marlin::Processor
init()
processRunHeader(LCRunHeader* run)
processEvent( LCEvent* evt)
check( LCEvent* evt)
end()
```

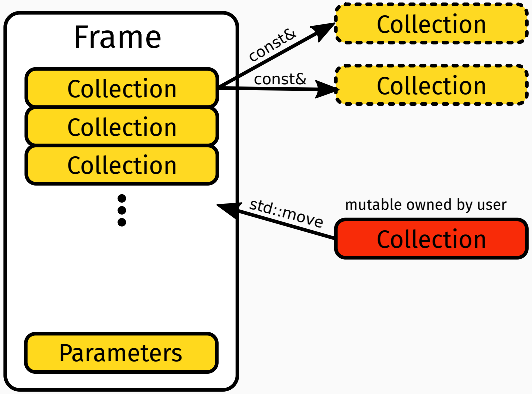
Schema evolution - Technical details

- Called as early as possible and as late as necessary
    - Earliest point where we have collection buffers from all backends is in Frame
- Schema evolution functions available from `SchemaEvolution` singleton
    - Populated during shared library loading
- Schema evolution can be a no-op


Frame
Map<string, unique_ptr<CollectionBase>> m_colls;
unique_ptr<FrameDataT> m_frameData;
Frame::get(std::string name)
in m_colls?
true
false
return collection;
auto buffers = m_frameData->getBuffers(name)
valid buffers
no buffers
buffers = schemaEvolution(buffers, buffers.version, buffers.type);
auto it = m_colls.emplace(name, buffers.create());
return it->second;
collection not present

The `Frame` - A generalized (event) data container

- *Type erased* container aggregating all relevant data
- Defines an *interval of validity* / category for contained data
  - Event, Run, readout frame, ...
- Easy to use and thread safe interface for data access
  - Immutable read access only
  - Ownership model reflected in API
- Decouples I/O from operating on the data



```cpp
template<typename CollT>
const CollT& get(const std::string& name) const;

template<typename CollT, /*enable_if*/>
const CollT& put(CollT&& collection,
                 const std::string& name);
```