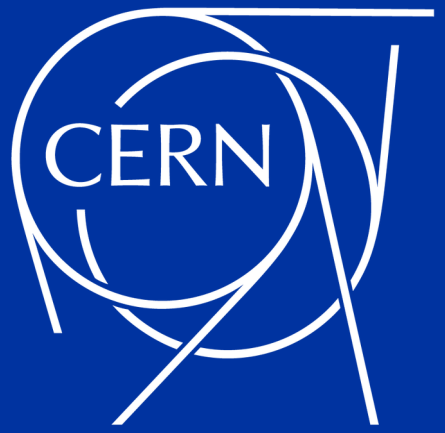


A New Model For FEC Hardware Description And Configuration

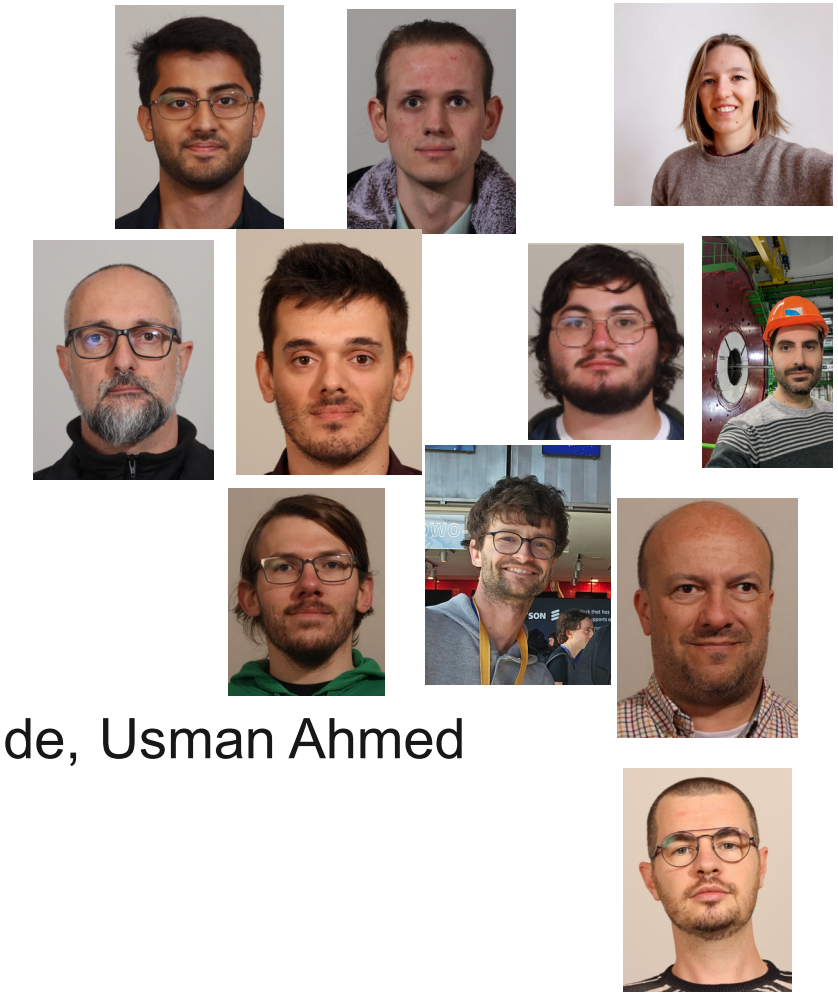
CCDE Release Candidate

Federico Vaga

2024-11-28

A Wide Collaborative Effort – BE-CEM / BE-CSS

- **CCDB**
  - Bartek Urbaniec, David Cobas, Federico Vaga
- **Data Migration**
  - Bartek Urbaniec, Federico Vaga, Jean-Francois Comblin
- **CCDA**
  - Stefano Gennaro, Mario Lombas
- **CCDE**
  - Anti Asko, Emanuele Matli, Emely Henninger, Raphael Wude, Usman Ahmed
- **Front-End Computer**
  - David Cobas, Federico Vaga, Orson Suminski



It Was Working! Why Did We Do This Work?


TECHNICAL DEBT
I DON'T UNDERSTAND WHY IT TAKES SO LONG TO ADD A NEW WINDOW.
@VINCENTDNL

Why Did We Do This Work? - Reflect Modern Reality

- **We want to clearly describe modern hardware**
  - The current architecture does not describe well modular or programmable devices
  - It will be possible to describe hardware connections with less error
    - More constraints, less conventions or oral traditions
- **We want to easily configure modern hardware**
  - The current architecture is designed around non-programmable hardware
  - It will be possible to assign configurations to hardware modules from CCDB
    - This includs binaries targeting FPGA, MCU, or memories and their drivers and versions
- **We want to be able to use modern software technologies**
  - The current architecture is based on software mainly developed around 1992 and unmaintained since 2015
  - Impossible to tame `dscinit`, the `transfer.ref` generator
  - The Linux ecosystem provides standard software that can be used to load drivers and configure hardware

Why Did We Do This Work? - New Possibilities

- **Implement DevOps techniques to handle entries in CCDB**
- **Create extensions toward secondary crates (coming in 2025)**
  - Describe hardware among different crate types (PCIe, PXIe, VME, MTCA, USB)
- **Cheby/EDGE generation from CCDB (coming in 2025)**
  - It will need a common effort (EDGE provider and users) to decide rules and policies
- ***Random list of other possibilities – no commitment to implement any of the following***
  - Run time comparison between CCDB declaration and real hardware
    - Including all binaries on hardware (FPGA, MCU, memories)
    - With limitations (many) on VME pre-VME64x
  - First time data entry in CCDB based on real hardware configuration
    - With limitations (many) on VME pre-VME64x
  - Running re-flashing campaign by changing CCDB declaration

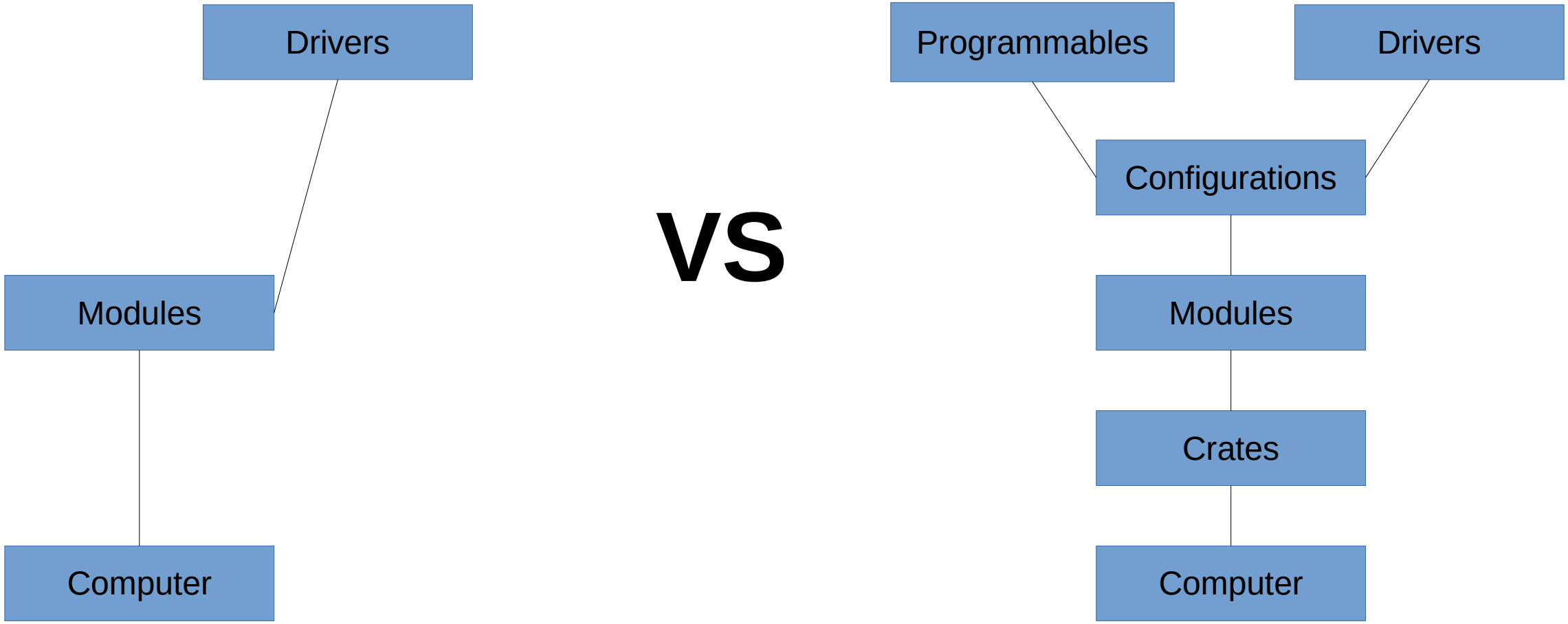
What Happens After Deployment? - CCDB/A/E

- **Deployment is foreseen for →→ Tuesday 21st January 2025 ←←**
- **Data will be migrated from the old CCDB schema to the new one**
  - The old data will be archived for reference
  - A few classes of modules will be subject to data manipulations
    - Some VFC-HD based systems, TXMC, MTCA
- **The new CCDA API will exit from its beta state**
  - You will be able to start using it: https://ccda.cern.ch:8900/api/swagger-ui/index.html
- **The new CCDE GUI will be promoted to production**
  - A very good entry point, it graphically represents all changes we have made
  - Play with it now and provide feedback: https://ccde-dev.cern.ch/

What Happens After Deployment? - FECs

- **Deployment is foreseen for →→ Tuesday 21st January 2025 ←←**
- **`fechw-tref-builder` replaces `dscinit` for `transfer.ref` generation**
- **The command `make transfer.ref` will stay in place, no changes for users**
- **We guarantee that the `transfer.ref` file does not change**
  - Manual changes remain manual changes – no support for that
- **Driver loading and hardware configuration will stay as it is today**
  - Next year we will focus on the `transfer.ref` eradication
- **Only LUMENS supported for the startup sequence**
  - Get in touch with Frank Locci if you still have systems not using LUMENS

Database – CCDx – Old versus New



Demonstration a Use Case – FIP From Scratch

01 Declare the Single Board Computer (SBC) type

- IPC647E

02 Declare the crate type

- IPC-2U-4PCIE (always embeds a IPC647E module)

03 Declare module types

- FMC-SPEC, CTRIE, FMC-MASTERFIP-1M

04 Declare driver types

- CTRP, fmc-spec, mockturtle

05 Declare Field Programmable Units

- The FMC-SPEC has one FPGA (Spartan-6)

06 Declare FPGA bitstream

- The spec-masterfip.bin can be used on a FMC-SPEC

07 Declare module configurations

- CTRP (CTRP)
- masterfip (fmc-spec, mockturtle, spec-masterfip.bin)

08 Create new computer

09 Create a new primary crate

10 Create module instances

- PCIe.01: FMC-SPEC + CTRIE
- PCIe.02: FMC-SPEC + FMC-MASTERFIP-1M

11 Assign configurations

- CTRP configuration to PCIe.01 FMC-SPEC
- masterfip configuration to PCIe.02 FMC-SPEC

12 Add Logical configuration

- It is very user dependent. Make up one

Extras

Hackers' Commands – Behind `make transfer.ref`

- **Today make `transfer.ref` eventually execute `dscinit <fec-hostname>`**
  - `dscinit` is a monolith doing everything
- **The new approach follows a pipe architecture**
  - Multiple tools process data in steps until the desired result
- **Source code: https://gitlab.cern.ch/be-cem-edl/fec/utilities/fec-hardware-description**
- **Users do not need to know about these tools nor use them**
- **But, if you are curious ...**

Hackers' Commands – Behind `make transfer.ref`

```
# Pre-requirement: prepare and activate a python virtual environment

git clone https://gitlab.cern.ch/be-cem-edl/fec/utilities/fec-hardware-description.git

cd fec-hardware-description

python3 -m pip install -U pip

python3 -m pip install -r requirements-build.txt

python3 -m pip install .

# Generate a new transfer.ref for a FEC of choice

export fec=cfc-774-cdv34

fechw-db-jsonsan --sql --input $fec.json --output ${fec}.sql

fechw-db-builder --schema src/fechw/data/hwdesc.sql --input-file $fec.sql --destdir $PWD $fec.db

fechw-tref-builder --database $fec.db --output-file $fec.tref $fec
```