Software development and deployment of the FCC Software

Javier Cervantes (CERN) for the FCC Software Team

FCC Week 2019, Brussels
26 June 2019
FCC Software in Context

- Built on top of Key4HEP stack
  - Common building blocks for future experiments
- Provide FCC-specific applications
- Integrate solutions into common stack system
- Sharing methodologies and tools
Pillars of Software Development

- Architecture, Patterns, Principles
- Testing
- Version Control System
FCC Estimated Timescale

- **Design**
- **Proto**
- **Construction**
- **Physics**
FCC Estimated Timescale

- Design: 2015 - 2025
- Proto: 2025 - 2035
- Construction: 2035 - 2040
- Physics: 2040 - 2050

25 years
FCC Estimated Timescale

Today
Core is ready

2010 2015 2020 2025 2030 2035 2040 2045 2050

- Design: ~25 years
- Proto: ~20 years
- Construction
- Physics
Challenges of writing SW for more than 20 years
Challenges of writing SW for more than 20 years

- Large number of people from different fields
Challenges of writing SW for more than 20 years

- Large number of people from different fields
- Wide range of skills and abilities
Challenges of writing SW for more than 20 years

- Large number of people from different fields
- Wide range of skills and abilities
- Large rotation of code authors (other projects, labs, industry...)
Challenges of writing SW for more than 20 years

- **Large number** of people from different fields
- **Wide range of skills and abilities**
- **Large rotation** of code authors (other projects, labs, industry...)
- **General solutions** for single-experiment scopes
Challenges of writing SW for more than 20 years

- Large number of people from different fields
- Wide range of skills and abilities
- Large rotation of code authors (other projects, labs, industry...)
- General solutions for single-experiment scopes

Future experiments’ software needs stable, robust and efficient supporting infrastructure
Building a project for decades
Software easy to use

Make interfaces easy to use correctly and hard to use incorrectly

Scott Meyers - The Most Important Design Guideline?
Software easy to use

Make interfaces easy to use correctly and hard to use incorrectly

Scott Meyers - The Most Important Design Guideline?

Simple to install
Software easy to use

Make interfaces easy to use correctly and hard to use incorrectly

Scott Meyers - The Most Important Design Guideline?

Simple to install

CONDA  Spack  Homebrew
Software easy to use

Make interfaces easy to use correctly and hard to use incorrectly

Scott Meyers - The Most Important Design Guideline?

Simple to install

Easy to update / maintain
Software easy to use

Make interfaces easy to use correctly and hard to use incorrectly

Scott Meyers - The Most Important Design Guideline?

Simple to install

Easy to update / maintain

Tutorials, documentation, training
Software easy to use

Make interfaces easy to use correctly and hard to use incorrectly

Scott Meyers - The Most Important Design Guideline?

Simple to install

Easy to update / maintain

Tutorials, documentation, training

Effective and clear rules for contributions
Continuous Integration and Deployment processes

Users → Feedback → New developments (bugs, features, refactor)

Code → Build → Integrate → Release → Deploy
Continuous Integration and Deployment processes

Users → Feedback → New developments
  (bugs, features, refactor)

Code → Build → Integrate → Release → Deploy

Continuous Integration
Continuous Integration and Deployment processes

Users → Feedback → New developments
(bugs, features, refactor)

Code → Build → Integrate → Release → Deploy

Continuous Integration

Jenkins  Travis CI  GitLab
Continuous Integration and Deployment processes

Users → Feedback → New developments (bugs, features, refactor)

- Code
- Build
- Integrate
- Release
- Deploy

Continuous Integration

Continuous Delivery
Continuous Integration and Deployment processes

Users → Feedback → New developments
(bugs, features, refactor)

- Code
- Build
- Integrate
- Release
- Deploy

Continuous Integration
Continuous Delivery

GitHub, Jenkins, Travis CI, GitLab, Docker
Continuous Integration and Deployment processes

Users → Feedback → New developments (bugs, features, refactor)

Code → Build → Integrate → Release → Deploy

Continuous Integration

Continuous Delivery

Continuous Deployment
Continuous Integration and Deployment processes

Users → Feedback → New developments
(bugs, features, refactor)

Code → Build → Integrate → Release → Deploy

Continuous Integration

Continuous Delivery

Continuous Deployment

Jenkins, docker, File system, puppet
Testing
Testing

What? Evaluate the quality of software
<table>
<thead>
<tr>
<th><strong>Testing</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What?</strong></td>
</tr>
<tr>
<td><strong>How?</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Testing

<table>
<thead>
<tr>
<th><strong>What?</strong></th>
<th>Evaluate the quality of software</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>How?</strong></td>
<td>Checking functional requirements</td>
</tr>
<tr>
<td></td>
<td>Forcing errors</td>
</tr>
<tr>
<td></td>
<td>Incrementally</td>
</tr>
<tr>
<td></td>
<td>Assessing usability and performance</td>
</tr>
<tr>
<td><strong>When?</strong></td>
<td><strong>Always</strong>, it is part of the process</td>
</tr>
<tr>
<td></td>
<td>New contributions, bugs, features</td>
</tr>
</tbody>
</table>
## Testing

<table>
<thead>
<tr>
<th><strong>What?</strong></th>
<th>Evaluate the quality of software</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>How?</strong></td>
<td>Checking functional requirements</td>
</tr>
<tr>
<td></td>
<td>Forcing errors</td>
</tr>
<tr>
<td></td>
<td>Incrementally</td>
</tr>
<tr>
<td></td>
<td>Assessing usability and <strong>performance</strong></td>
</tr>
<tr>
<td><strong>When?</strong></td>
<td><strong>Always</strong>, it is part of the process</td>
</tr>
<tr>
<td></td>
<td>New contributions, bugs, features</td>
</tr>
<tr>
<td><strong>Why?</strong></td>
<td>Quality</td>
</tr>
<tr>
<td></td>
<td>Faster Feedback</td>
</tr>
<tr>
<td></td>
<td>Efficiency</td>
</tr>
<tr>
<td></td>
<td>Earlier Error detection</td>
</tr>
<tr>
<td></td>
<td>User experience</td>
</tr>
<tr>
<td></td>
<td>Longer project life</td>
</tr>
</tbody>
</table>
Good Practices
Good Practices

- Effective coding techniques
  - Evolution and maintainability of the code
  - Adopt new programming models and technologies without breaking API’s
Good Practices

▷ Effective coding techniques
  ○ Evolution and maintainability of the code
  ○ Adopt new programming models and technologies without breaking API’s

▷ Version control systems
  ○ Mandatory for reproducibility and deployments

GitHub
GitLab
Good Practices

▸ **Effective coding techniques**
  ○ Evolution and maintainability of the code
  ○ Adopt new programming models and technologies without breaking API’s

▸ **Version control systems**
  ○ Mandatory for reproducibility and deployments

▸ **Community engagement**
  ○ Help users and *developers* (Forums, documentation, ...)
  ○ Examples, tutorials, support
Good Practices

▸ Effective coding techniques
  ○ Evolution and maintainability of the code
  ○ Adopt new programming models and technologies without breaking API's

▸ Version control systems
  ○ Mandatory for reproducibility and deployments

▸ Community engagement
  ○ Help users and developers (Forums, documentation, ...)
  ○ Examples, tutorials, support

▸ Sharing the codebase between programmers
  ○ Code reviews, design discussions
Current status
FCC Software today

- **Two main deliverables:**
  - **FCCSW:** FCC software, framework common to FCC-hh, -ee, and -eh
  - **Externals:** FCC-specific software dependencies

- **Computing resources**
  - Shared with LCG infrastructure
  - CERN Openstack virtual machines + LCG Physical nodes
  - CVMFS as main software repository for distribution

- **Build services based on** [Spack](http://spack.readthedocs.io)
  - Automated with Jenkins

**Setup the FCC environment**

```
source /cvmfs/fcc.cern.ch/sw/views/releases/externals/94.2.0/x86_64-centos7-gcc62-opt/setup.sh
```
# Common conventions

<table>
<thead>
<tr>
<th>Adopted community guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency</td>
</tr>
<tr>
<td>Interoperability</td>
</tr>
<tr>
<td>Efficiency</td>
</tr>
</tbody>
</table>
HEP Software Foundation Project templates

Documentation

Code structure

Similar packaging

Extensible to python modules or code techniques

Common HSF Tools

### PROJECTTEMPLATE

Please add some lines describing the project!

#### Building the project

```
mkdir build
cd build
cmake .DMAKE_INSTALL_PREFIX=<installdir> [DDPROJECTTEMPLATE_BUILD_DOCS=ON] <path to sources>
make -j <number of cores on your machine>
make install
```

The `PROJECTTEMPLATE_BUILD_DOCS` variable is optional, and should be passed if you wish to build the Doxygen based API documentation. Please note that this requires an existing installation of Doxygen. If CMake cannot locate Doxygen, its install location should be added into `CMAKE_PREFIX_PATH`. For further details please have a look at the CMake tutorial.

#### Building the documentation

The documentation of the project is based on doxygen. To build the documentation, the project must have been configured with `PROJECTTEMPLATE_BUILD_DOCS` enabled, as described earlier. It can then be built and installed:
HSF Project templates - Example

```cpp
#include(GNUInstallDirs)

# Define basic build settings
# Use GNU-style hierarchy for installing build products

if(NOT CMAKE_BUILD_TYPE)
  set(CMAKE_BUILD_TYPE RelWithDebInfo)
  CACHE STRING "Choose the type of build, options are: None Release MinSizeRel Debug RelWithDebInfo" FORCE
  else()
    set(CMAKE_BUILD_TYPE "${CMAKE_BUILD_TYPE}")
    CACHE STRING "Choose the type of build, options are: None Release MinSizeRel Debug RelWithDebInfo" FORCE
  endif()
endif()
```

CMake Project Template

FCC-EDM Package
Community-oriented mindset

FCC Applications

LCG / Key4HEP
Stack system

OS Kernel and Libraries
(Non-HEP specific)
Community-oriented mindset

Benefit from the **common efforts**

- **LCG / Key4HEP Stack system**
- **FCC Applications**
- **OS Kernel and Libraries (Non-HEP specific)**
Community-oriented mindset

- Benefit from the common efforts
- Provide feedback to the community
Community-oriented mindset

- Benefit from the common efforts
- Provide feedback to the community
- Build software aiming to contribute
Community-oriented mindset

- Benefit from the **common efforts**
- Provide **feedback** to the community
- Build software **aiming to contribute**
- Documentation, forums, support

---

FCC Applications

- LCG / Key4HEP Stack system

- OS Kernel and Libraries (Non-HEP specific)
FCC Software Website

Documentation, tutorials, infrastructure

FCCSW

Software for the Future Circular Collider.

About

FCCSW is a set of software packages, tools, and standards to help different FCC studies work together. Common software helps to avoid duplicated effort and compare results. In addition, the software group provides infrastructure and services such as build systems, testing and continuous integration, code format guidelines, linting and static analysis, release management and software distribution and data persistency. This is possible due to the kind support of the EP-SFT group.

Conceptual Design Report

http://hep-fcc.github.io/FCCSW/
FCC Software Forum

Users support

https://fccsw-forum.web.cern.ch

FCCSW
Discuss anything related to the FCC Software here. Please post bug reports in Jira

Site Feedback
Discussion about this site, its organization, how it works, and how we can improve it.

Uncategorized
Topics that don’t need a category, or don't fit into any other existing category.
FCC Software Jira

https://sft.its.cern.ch/jira/projects/FCC/issues

Issue tracker

[Image of Jira interface showing open issues and details of an issue titled "PODIO Reader need to support chain of root files"]]
Providing software to users

- Software stacks need to be made available to users
  - In parallel to the development process
- Covering different configurations
  - Compilers, platforms, architectures, stack versions
- Stable and bleeding edge versions
  - Releases (static), Nightlies (ephemeral, likely unstable)
- Flexible to cover different use-cases
  - Production, grid jobs, developments, testing
Providing software - *Current approach*

- **Package A**
- **Package B**
- **Package C**
- **Package Nth**

**Compiled and installed**

**Software Distribution system - CVMFS**

- **Grid Jobs**
- **User**
- **Data Analyst**
- **Software developer**

https://github.com/HEP-FCC/fcc-spi
Tooling for users
Tooling for users

- Prepare full or partial environment
Tooling for users

- Prepare **full or partial environment**
  - All the packages to run data analysis
  - Set up all the dependencies to develop a package
Tooling for users

▸ Prepare full or partial environment
▸ Select and switch from different stack versions
Tooling for users

- Prepare full or partial environment
- Select and switch from different stack versions

Grid Jobs

User

Data Analyst

... Software developer

Jenkins

Testing, CI

Production Old stable release Release Candidate Nightlies
Tooling for users

- Prepare full or partial environment
- Select and switch from different stack versions
- Hide complexity
Tooling for users

- Prepare **full or partial environment**
- Select and switch from **different stack versions**
- **Hide complexity**
- **Reproducibility of environment**
Tooling for users

- Prepare full or partial environment
- Select and switch from different stack versions
- Hide complexity
- Reproducibility of environment

Not yet there, looking into existing experiments workflows (e.g. LHCb)
Conclusions

▸ **Common base** for future the experiments: *Key4HEP stack system*

▸ **Community-oriented mindset:**
  - Build generic tools useful for similar experiments / technologies
  - Contribute to the common layer

▸ **Rely on** stable, robust, maintained and efficient software

▸ **Developers**
  - Follow good practices: testing, coding techniques, agile development cycles
  - Follow community guidelines

▸ **Users**
  - Provide them with tooling to effectively handle software stacks
Thank you for your attention