

# THE EVOLUTION OF THE DOOCS C++ CODE BASE

L. Fröhlich\*, A. Aghababayan, S. Grunewald, O. Hensler, U.F. Jastrow, R. Kammering, H. Keller, V. Kocharyan, M. Mommertz, F. Peters, A. Petrosyan, G. Petrosyan, L. Petrosyan, V. Petrosyan, K. Rehlich, V. Rybnikov, G. Schlesselmann, J. Wilgen, T. Wilksen, DESY, Hamburg, Germany

## Modernization of Code and Development Techniques

Preparing DOOCS for future projects

**Goals**

Improve:  
readability  
maintainability  
stability  
teamwork

**Build System**  
Move from make to Meson

**Base Library**  
New General Utility Library for C++14 (GUL14)

**Training**  
Series of DOOCS lectures  
C++ style guide

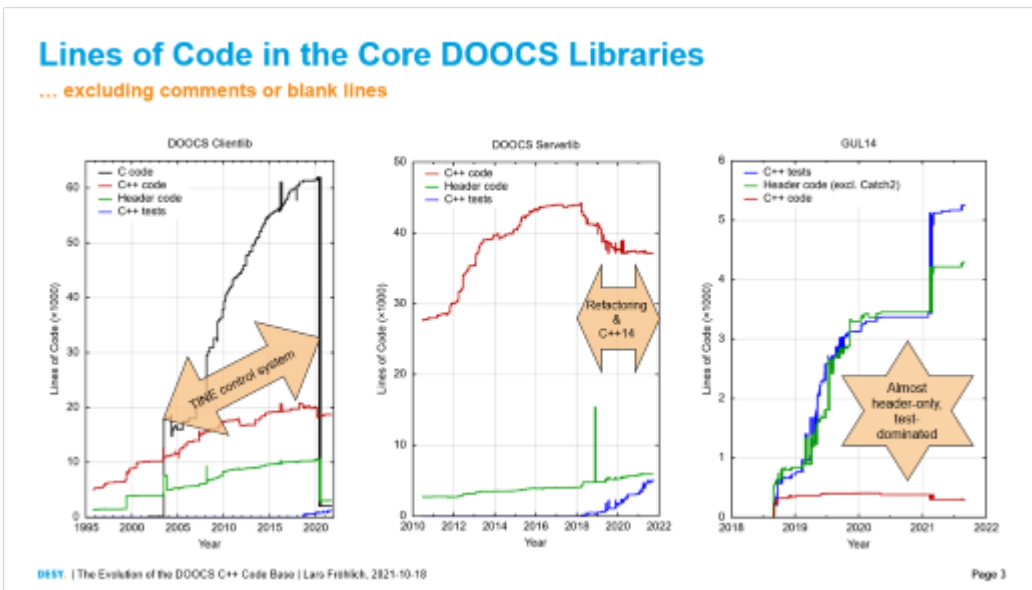
**Code Review**  
Required for core libs  
Encouraged for other libs  
Optional for servers

**Unit Tests**  
Add tests for old code (before refactoring)  
Refactor APIs to allow testing  
Catch2 unit testing framework

**Continuous Integration**  
Move to in-house Gitlab for DevOps

**Cross-Project Refactoring**  
API breaks require global modifications  
Access to almost all repositories

Page 2



## GUL14

General Utility Library for C++14

**Base library** (think Abseil [Google] or Folly [Facebook], but smaller)

**Open source** (LGPLv2.1, <https://www.desy.de/mcs/docs/gul/>)

**Multi-platform:** Linux, MacOS, Windows

Code of **wide applicability**

**No external dependencies** except C++/C standard libraries

**No control system** specific code

**Quality standards:**

- Style:** Code must follow our C++ style guide and should follow the C++ Core Guidelines
- Documentation:** Every function, type, etc. must be documented
- Unit tests:** Every entity in the library must have associated tests
- Code review:** Every commit must be approved by at least one other developer

Page 4

## DOOCS Fact Sheet

Some facts and figures around the control system

**1992:** DOOCS is born as a control solution for vacuum devices for superconducting cavity test stands at what will become the Tesla Test Facility **TF**. It soon gets ported the **HERA**

Today, DOOCS is used at the particle accelerators **ARES**, **European XFEL**, **FLASH**, **PETRA III**, and numerous smaller facilities at DESY. There are a few external users as well, proton storage ring.

In the '90s, object-oriented programming becomes a hot trend in software development. DOOCS stands for "Distributed Object-Oriented Control System". To efficiently control hardware and use high-level abstractions, **C++** is the natural choice.

DOOCS is built around the **SunRPC/ONC RPC** remote procedure call which is better known as the protocol behind the network file system **NFS**.

~3 **core libraries:** GUL14, clientlib, serverlib

~100 **other libraries:** hardware support, DAQ, databases, high-level controls, particle tracking, ...

~500 **server types:** connect hardware devices, process/archive data, run feedback loops, evaluate data, execute advanced algorithms

~8000 C++ source files

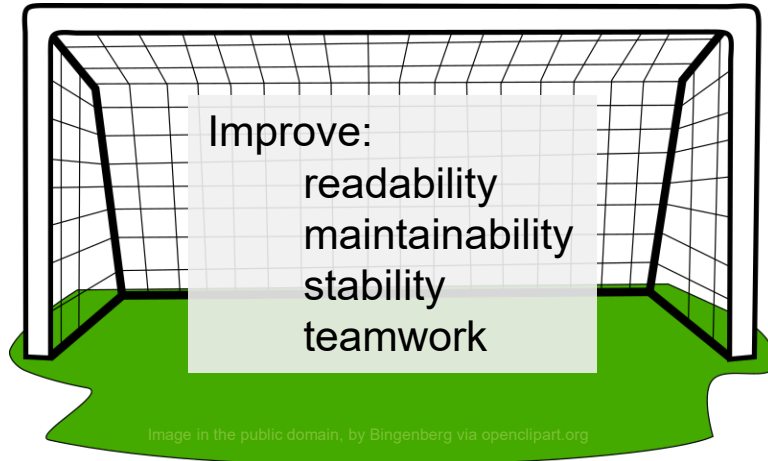
~1.5 million lines of code

Page 5

# Modernization of Code and Development Techniques

## Preparing DOOCS for future projects

### Goals



#### Build System

Move from make to Meson

#### Base Library

New General Utility Library for C++14 (GUL14)

#### Training

Series of DOOCS lectures  
C++ style guide

#### Unit Tests

Add tests for old code (before refactoring)  
Refactor APIs to allow testing  
Catch2 unit testing framework

#### Code Review

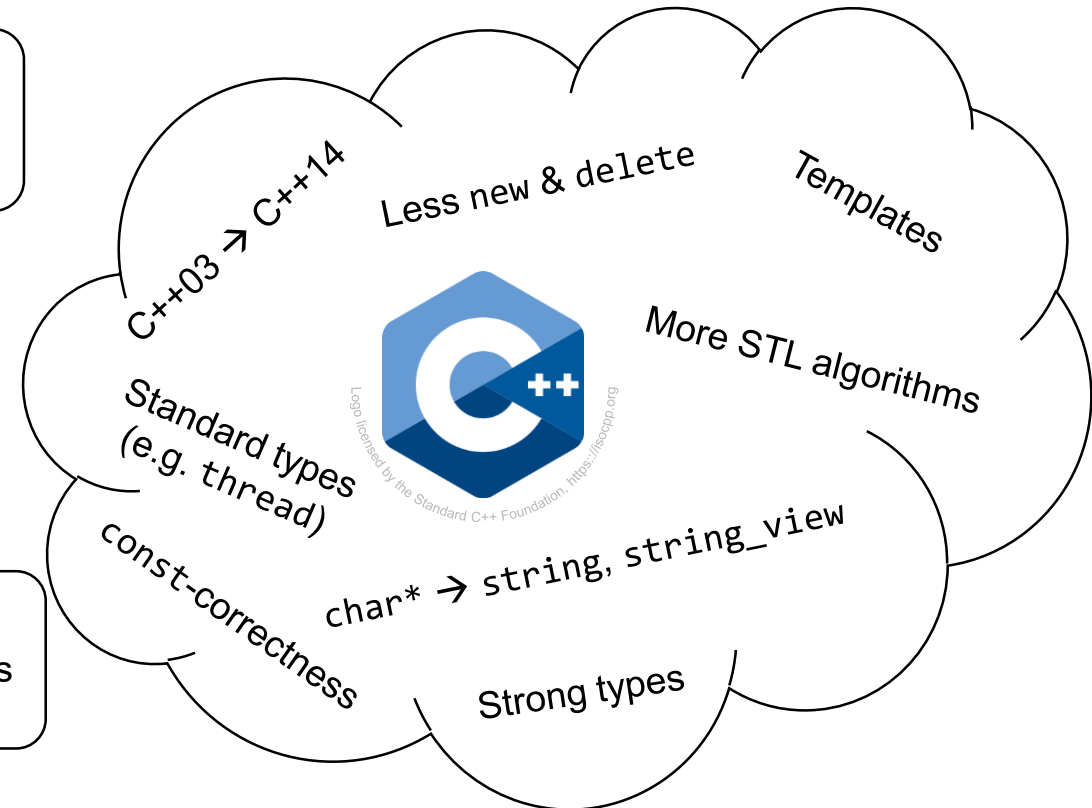
Required for core libs  
Encouraged for other libs  
Optional for servers

#### Continuous Integration

Move to in-house Gitlab for DevOps

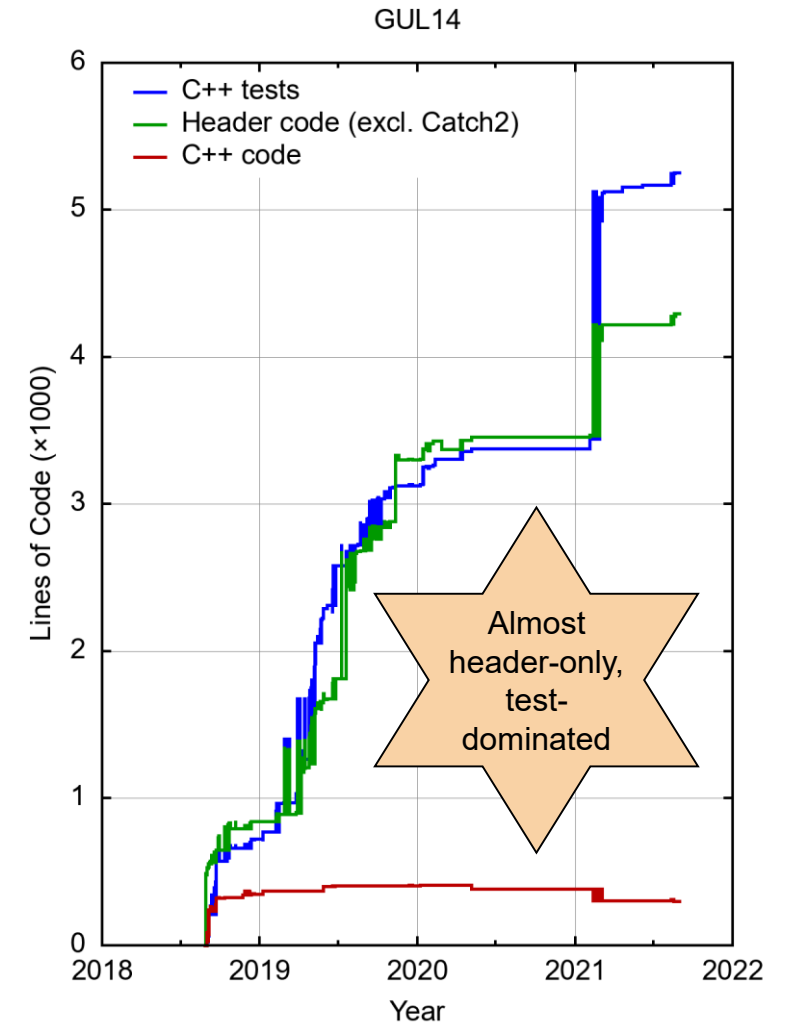
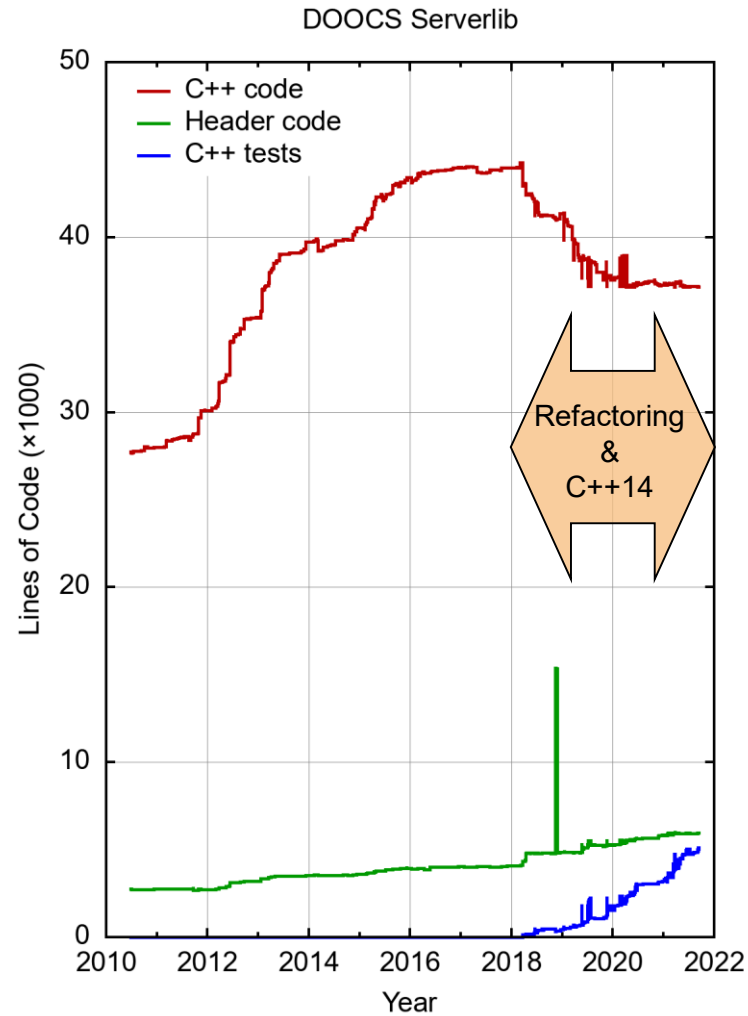
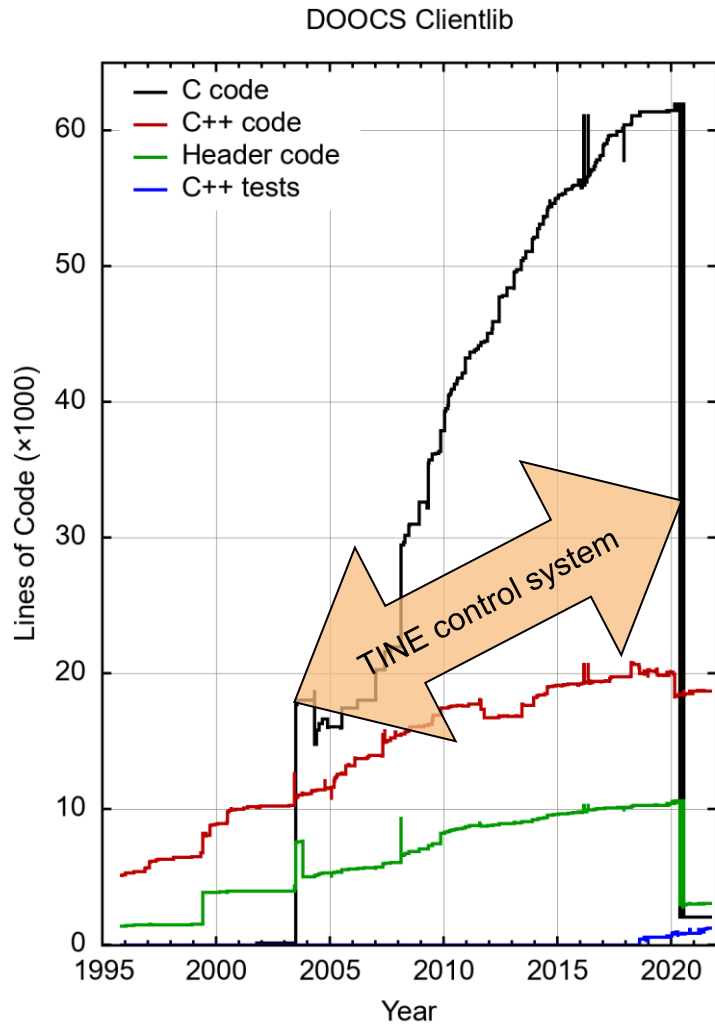
#### Cross-Project Refactoring

API breaks require global modifications  
Access to almost all repositories



# Lines of Code in the Core DOOCS Libraries

... excluding comments or blank lines



# GUL14

## General Utility Library for C++14

**Base library** (think Abseil [Google] or Folly [Facebook], but smaller)

**Open source** (LGPLv2.1, <https://winweb.desy.de/mcs/docs/gul/>)

**Multi-platform:** Linux, MacOS, Windows

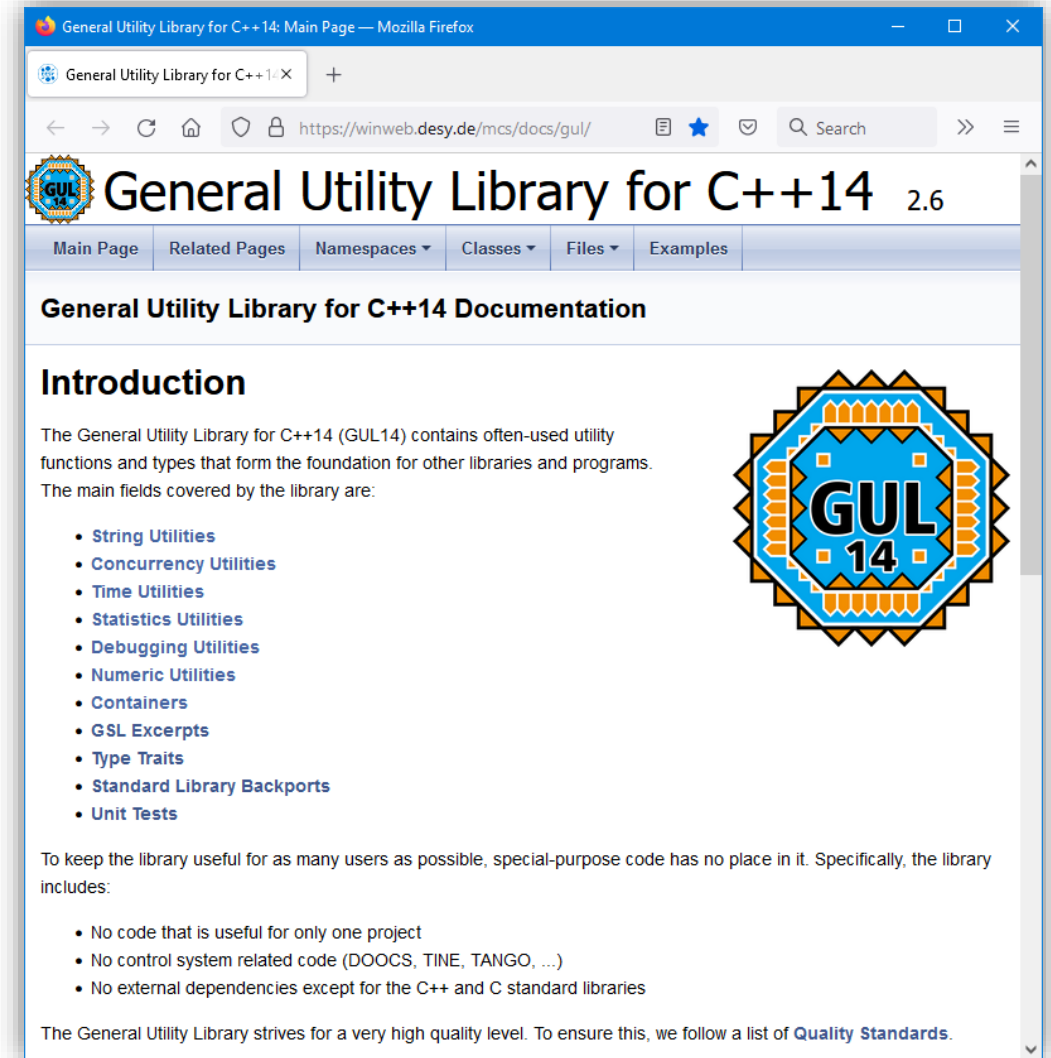
Code of **wide applicability**

**No external dependencies** except C++/C standard libraries

**No control system** specific code

Quality standards:

- **Style:** Code must follow our C++ style guide and should follow the C++ Core Guidelines
- **Documentation:** Every function, type, etc. must be documented
- **Unit tests:** Every entity in the library must have associated tests
- **Code review:** Every commit must be approved by at least one other developer



General Utility Library for C++14: Main Page — Mozilla Firefox

General Utility Library for C++14

General Utility Library for C++14 2.6

Main Page Related Pages Namespaces Classes Files Examples

### General Utility Library for C++14 Documentation

#### Introduction

The General Utility Library for C++14 (GUL14) contains often-used utility functions and types that form the foundation for other libraries and programs. The main fields covered by the library are:

- String Utilities
- Concurrency Utilities
- Time Utilities
- Statistics Utilities
- Debugging Utilities
- Numeric Utilities
- Containers
- GSL Excerpts
- Type Traits
- Standard Library Backports
- Unit Tests

To keep the library useful for as many users as possible, special-purpose code has no place in it. Specifically, the library includes:

- No code that is useful for only one project
- No control system related code (DOOCS, TINE, TANGO, ...)
- No external dependencies except for the C++ and C standard libraries

The General Utility Library strives for a very high quality level. To ensure this, we follow a list of [Quality Standards](#).

# DOOCS Fact Sheet

## Some facts and figures around the control system

**1992:** DOOCS is born as a control solution for vacuum devices for superconducting cavity test stands at what will become the Tesla Test Facility **TTF**. It soon gets ported to the **HERA** proton storage ring.

In the '90s, object-oriented programming becomes a hot trend in software development. DOOCS stands for “**Distributed Object-Oriented Control System**”. To efficiently control hardware and use high-level abstractions, **C++** is the natural choice.

DOOCS is built around the **SunRPC/ONC RPC** remote procedure call which is better known as the protocol behind the network file system NFS.

Today, DOOCS is used at the particle accelerators **ARES**, **European XFEL**, **FLASH**, **PETRA III**, and numerous smaller facilities at DESY. There are a few external users as well.



Images: Copyright DESY 2015, 2017

**3 core libraries:** GUL14, clientlib, serverlib

**~100 other libraries:** hardware support, DAQ, databases, high-level controls, particle tracking, ...

**~500 server types:** connect hardware devices, process&archive data, run feedback loops, evaluate data, execute advanced algorithms

**~8000** C++ source files

**~1.5 million** lines of code