# Julia for large HEP projects?

"I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation. We still make syntax errors, to be sure; but they are fuzz compared with the conceptual errors in most systems. If this is true, building software will always be hard.
**There is inherently no silver bullet.**"
[F. P. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering", IEEE Computer, Vol. 20, No. 4 (1987) 10]

Stefan Kluth
MPI für Physik
Julia for HEP Mini-workshop
27 Sep 2021

## Introduction to discussion

- Whoami?
  - Speak Fortran, C++, python, but not (yet) julia
  - OPAL, BaBar, ATLAS, JADE (re-analysis) analyses, S&C
  - Know about and teach OOP, clean code (a la uncle Bob), unit testing, refactoring, ...
- Julia is founded on multiple-dispatch
  - Implications for “analysis and design” of large software systems? It is different on purpose ...

# Aspects of big projects

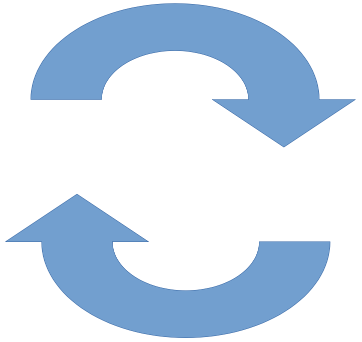
- Long lifetime, large codebase, many components
  - Fortran(77) stretched beyond O(100k) loc
  - C++ and OOP: clear interfaces, replaceable core components
  - Athena, cmssw, geant4, root
- Most of the work spent in maintenance mode
  - Bug fixes, enhancements, ports
  - Increasing difficulty to keep code “clean”

## Code, test, refactor

**Write code**

new feature, bug fix, etc

Efficient build system, scm (git)



**Refactoring**

Adapt bad interfaces, new platforms, efficiency bottlenecks, ugly code, etc

IDE support, “language server”, low language complexity

**Unit / integration tests**

protect existing code, prove new code

Libraries: gtest, pytest, Junit, ...

# Today?

- C++ hard to refactor, IDE support incomplete
  - In HEP, unit test coverage low, w/o unit test libraries
  - Python easier, but unit test coverage?
- Parts of code base difficult to maintain
  - Athena, geant4, root, …
- Analyses w/o “big frameworks”
  - Multiple small frameworks, “simple” intermediate data
  - Trend points to problems with exp frameworks

## Strawperson questions

- E.g. geantV in Julia?
  - Physics and algorithms known, implementation “easy”, but maintenance, upgrades, ports?
- Online vs offline algorithms (2nd level software trigger)
  - Online: real time app vs GC
- JIT compilation vs shared libraries: deployment?
- Software engineering know-how for julia?
  - Relationship to (partially) known OOP paradigms?
  - Large investment in OOP knowledge wasted or useful?