Contribution ID: **156** Type: **Oral**

ROOT and new programming paradigms

Monday, 10 October 2016 14:15 (15 minutes)

ROOT is one of the core software tool for physicists. For more than a decade it has a central position in the physicists' analysis code and the experiments' frameworks thanks in parts to its stability and simplicity of use. This allowed software development for analysis and frameworks to use ROOT as a "common language" for HEP, across virtually all experiments.

Software development in general and in HEP frameworks in particular has become increasingly complex over the years. From straightforward code fitting in a single FORTRAN source file, HEP software has grown over the years to span millions of lines of code spread amongst many, more or so collaborating, packages and libraries. To add to the complexity, in an effort to better exploit current and upcoming hardware, this code is being adapted to move from purely scalar and serial algorithm to complex multithread, multi-tasked and/or vectorized versions.

The C++ language itself and the software development community's understanding of the best way to leverage its strength has evolved significantly. One of the best example of this being the "C++ Core Guidelines"which purports to get a "smaller, simpler and safer language"out of C++. At the same time new tools and techniques are being developed to facilitate proving and testing the correctness of software programs, as exemplified by the C++ Guideline Support Library, but those require the tool to be able to understand the semantic of the interfaces. Design patterns and interface tricks that were appropriate in the early days of C++ are often no longer the best choices for API design. ROOT is at the heart of virtually all physics analysis and most HEP frameworks and as such needs to lead the way and help demonstrate and facilitate the application of those modern paradigms.

This presentation will review what theses lessons are and how they can be applied to an evolution of the ROOT C++ interfaces, striking a balance between conserving familiarity with the legacy interfaces (to facilitate both the transition of existing code and the learning of the new interfaces) and significantly improving the expressiveness, clarity, (re)usability, thread friendliness, and robustness of the user code.

Tertiary Keyword (Optional)

High performance computing

Secondary Keyword (Optional)

Software development process and tools

Primary Keyword (Mandatory)

Analysis tools and techniques

Primary author: CANAL, Philippe (Fermi National Accelerator Lab. (US))

Co-author: NAUMANN, Axel (CERN)

Presenter: CANAL, Philippe (Fermi National Accelerator Lab. (US))

Session Classification: Track 5: Software Development

Track Classification: Track 5: Software Development