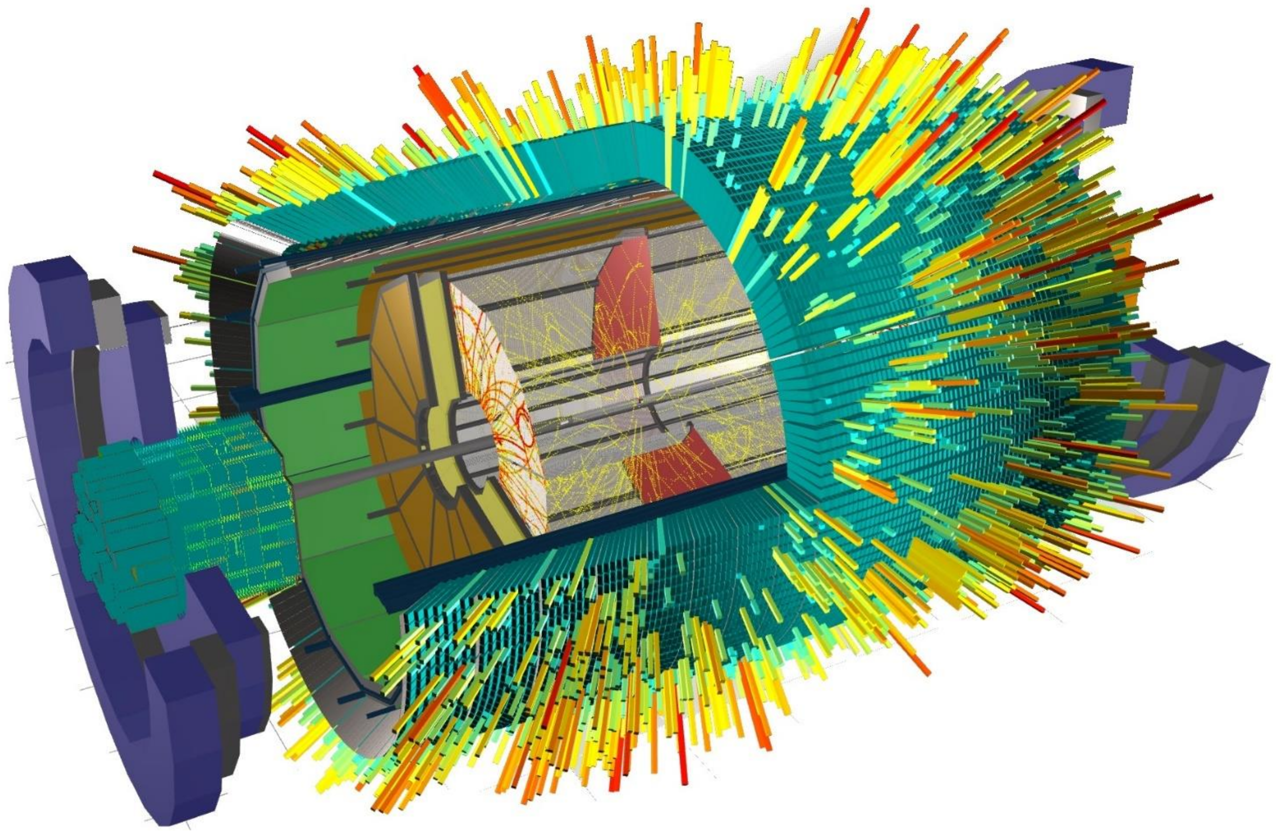
Application of SD Best Practices for the MPD Experiment

HNATIC Slavomir
MPD Software Development Team



OUTLINE

- Initial Status: Summer 2021 (Analysis)
- SD Best Practices
- Software vs R&D
- Software Project Dynamics
- Scaling and Complexity
- Unified Development Environment, Build & Software distribution system
- Design by Contract
- Future - MPD Data Lab
- Acceptance TDD
- Rapid Development
- MPD Software & Computing Ecosystem – The Big Picture

INITIAL STATUS (as of summer 2021)

Some of the most important findings:

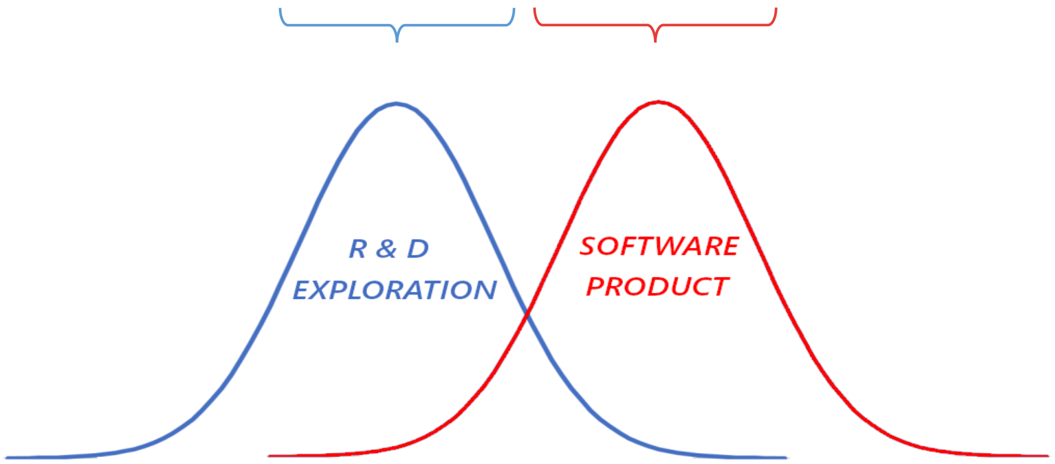
- Total lack of staff
- No code influx control (reviews)
- Lack of tests
- Dead/untested code hanging all around the place, its maintenance taking away from little worktime (man-hours) we have
- No OO code
- Codebase: one giant tightly coupled “global state/god class antipatterns” blob
- Cumbersome error-prone installation procedure
- Outdated website
- Lack of support & proper interaction with users, almost no user feedback

SD BEST PRACTICES

*"…the profession in which a knowledge of the mathematical and natural sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize, economically, the materials
and forces of nature for the benefit of mankind."*

-- Accreditation Board of Engineering & Technology (www.abet.org)

Technology Development =
Scientific Theory + Engineering Practice + Economy



ACTIVITIES FOCUS

SEPARATION OF CONCERNS

- thinking of software entity attributes in isolation, while keeping in mind, they're part of the whole

E.Dijkstra "On the role of scientific thought" (1974)

CORE INFLUENCES

- size / scaling
- structural complexity
- software defects
- uncertainty
- human variation
- synergy

SWEBOK v3 (2015, computer.org)
International ISO Standard
specifying the guide to
Software Engineering Body of Knowledge

R&D vs SOFTWARE

R & D

CONCEPT VALIDITY EXPLORATION

- Key goal: Innovation
- Successful end justifies all means
- Many of tested hypotheses invalid
- Proper practices completely out of focus to save time
- Prototypes of valid concepts must be adapted to SE standards

SOFTWARE ENGINEERING

PRODUCT DEVELOPMENT

- R&D valid concepts integrated into whole
- Not in conflict with existing development
- User/developer friendliness
- Extensible
- Maintainable
- Not requiring unmanageable (geeky) support
- Compact, modular
- Follows SE principles & best practices

CODE INFLUX CONTROL - CODEOWNERS

CODEOWNERS 1.81 KB

```
# Allowed line format:
# 0. Empty and commented lines
# 1. <path ending with />  @owner_1 @owner_2     @owner_N
#    Example: /scripts/ @roleg @hnatics
#    Ending a path in a `/` will specify the code owners for every file nested in that directory,

# Default (project) owner
#---------------------------------------------------------------------------------
/                                       @roleg


# System
#---------------------------------------------------------------------------------
/scripts/                               @hnatics


# Detectors
#---------------------------------------------------------------------------------
# TPC
/detectors/tpc/                         @zinchenk  @abychkov

# FFD
/detectors/ffd/                         @nlashmanov

# EMC
/detectors/emc/                         @boiana2012

# MCORD
/detectors/mcord/                       @Wielanek

# STS  (ITS)
/detectors/sts/                         @vkondrat

# TOF
/detectors/tof/                         @lobastov

# ZDC  (FHCAL)
/detectors/zdc/                         @marina


# Reconstruction
#---------------------------------------------------------------------------------
/reconstruction/tracking/kalman/        @zinchenk

/core/mpdPid/                           @zinchenk @amudrokh


# Analysis
#---------------------------------------------------------------------------------
/core/mpdDst/MpdMiniEvent/              @gnigmat


# Database
#---------------------------------------------------------------------------------
/tools/database/                        @busa


# Documentation
#---------------------------------------------------------------------------------
/tools/documentation/                   @busa
```

"The art of programming (software development) is the art of organizing complexity, of mastering multitude and avoiding its bastard chaos as effectively as possible."
E. Dijkstra

Code ownership within Gitlab

- forces assignment of responsibilities
- automatically checks for ownership of changed files
- emails owners asking them for a review

CodeOwners Admin @project_24_bot · 2 months ago Maintainer

Dear @roleg, @hnatics, you have been identified as code owner of at least one file which was changed with this merge request. Please check the changes and approve them or request changes.

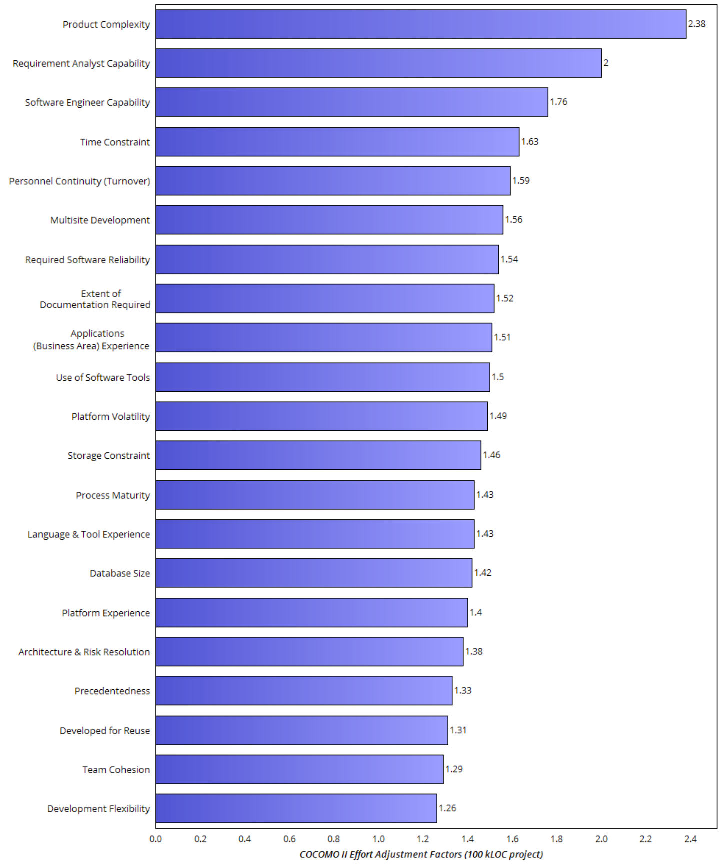
Effect

- code review by competent developers
- no arbitrary merges, trash code influx halted
- split between R&D and software code

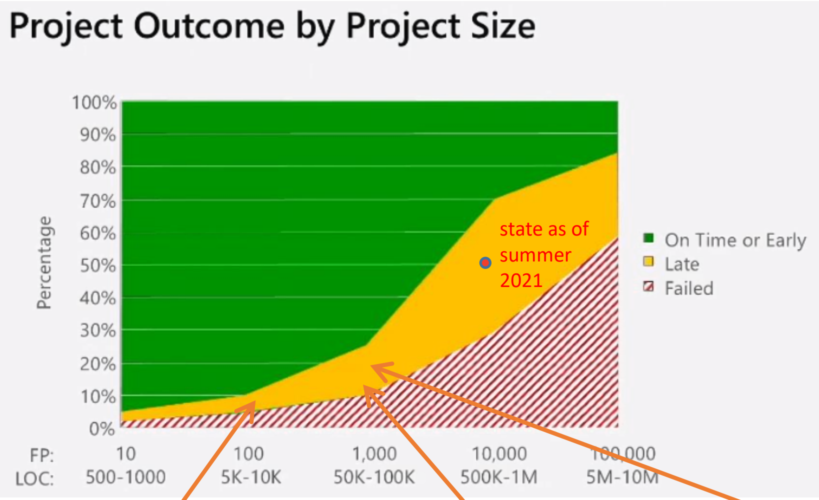
SOFTWARE PROJECT DYNAMICS

COnstructive COst MOdel (COCOMO II) by Barry W. Boehm

- Most rigorous statistical analysis of software projects using data from historic projects
- Results expressed in “effort adjustment factors”, these describe software project dynamics, used to gain insight to adjust the development strategy
- Requirements Analyst Capability factor 2 means project with very low level analysis of requirements would cost 2 times more effort than project with very high level of requirements analysis



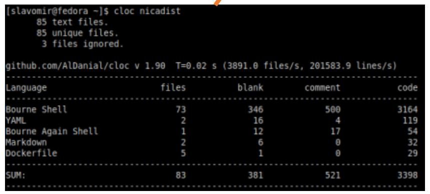
SCALING & COMPLEXITY REDUCTION



Applied Software Measurement, C. Jones (2008)

Some of the major reasons for dysfunctional scaling:

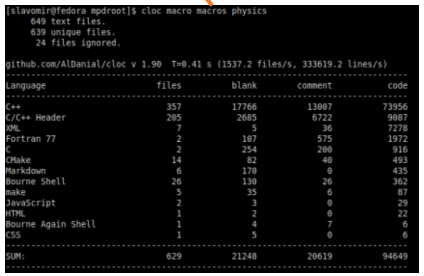
- Building on a weak foundation (overall SD setup, SoC - code restructuring, decoupling)
- Lack of proper technical practices (testing, TDD, reviews, documentation, OOA/OOD)
- Weak product & user level focus (release schedule, user feedback)
- Unused code hanging all over the place (code influx control, cleanup)
- Lack of direction (big picture view, milestones, prioritization)



```
[slavomir@fedora ~]$ cloc nicadist
      85 text files.
      85 unique files.
       3 files ignored.

github.com/AlDanial/cloc v 1.90  T=0.02 s (3891.0 files/s, 201583.9 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Bourne Shell                    73            346            500           3164
YAML                             2             16              4            119
Bourne Again Shell               1             12             17             54
Markdown                         2              6              0             32
Dockerfile                       5              1              0             29
-------------------------------------------------------------------------------
SUM:                            83            381            521           3398
-------------------------------------------------------------------------------
```

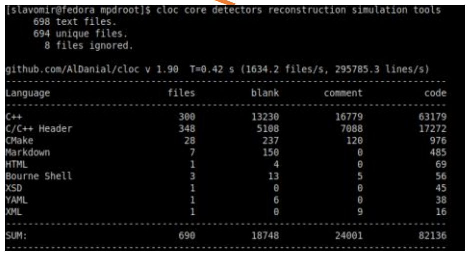
Build



```
[slavomir@fedora mpdroot]$ cloc macro macros physics
     649 text files.
     639 unique files.
      24 files ignored.

github.com/AlDanial/cloc v 1.90  T=0.41 s (1537.2 files/s, 333619.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                            357          17766          13007          73956
C/C++ Header                   205           2685           6722           9087
XML                              7              5             36           7278
Fortran 77                       2            107            575           1972
C                                2            254            200            916
CMake                           14             82             40            493
Markdown                         6            170              0            435
Bourne Shell                    26            130             26            362
make                             5             35              6             87
JavaScript                       2              3              0             29
HTML                             1              2              0             22
Bourne Again Shell               1              4              7              6
CSS                              1              5              0              6
-------------------------------------------------------------------------------
SUM:                           629          21248          20619          94649
-------------------------------------------------------------------------------
```

Physics



```
[slavomir@fedora mpdroot]$ cloc core detectors reconstruction simulation tools
     698 text files.
     694 unique files.
       8 files ignored.

github.com/AlDanial/cloc v 1.90  T=0.42 s (1634.2 files/s, 295785.3 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                            300          13230          16779          63179
C/C++ Header                   348           5108           7088          17272
CMake                           28            237            120            976
Markdown                         7            150              0            485
HTML                             1              4              0             69
Bourne Shell                     3             13              5             56
XSD                              1              0              0             45
YAML                             1              6              0             38
XML                              1              0              9             16
-------------------------------------------------------------------------------
SUM:                           690          18748          24001          82136
-------------------------------------------------------------------------------
```

Backend

CODE RESTRUCTURING & CLEANUP

Top Level

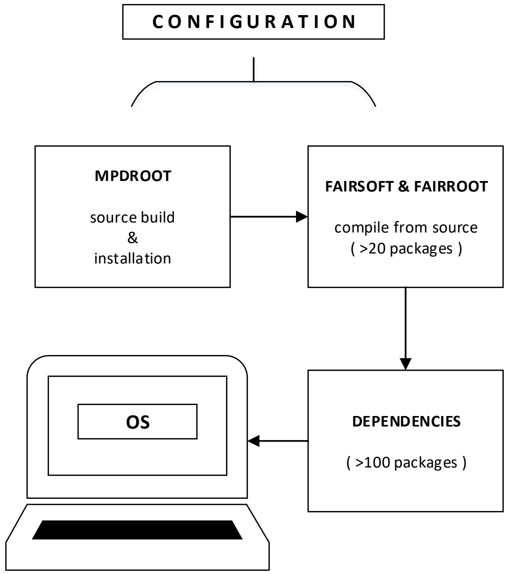
40 directories (1y ago) --> 14 directories (now)

- Unused detectors removal
- Old dysfunctional test system replaced
- Junk files removal (old scripts, configs, styling)
- Unused libraries removal
- Deployment system replaced & decoupled

SCALING: indicates action of cumulative forces pushing projects towards either success or failure

BUILD & SOFTWARE DISTRIBUTION SYSTEM

BEFORE: OVERWHELMING COMPLEXITY (for every user)

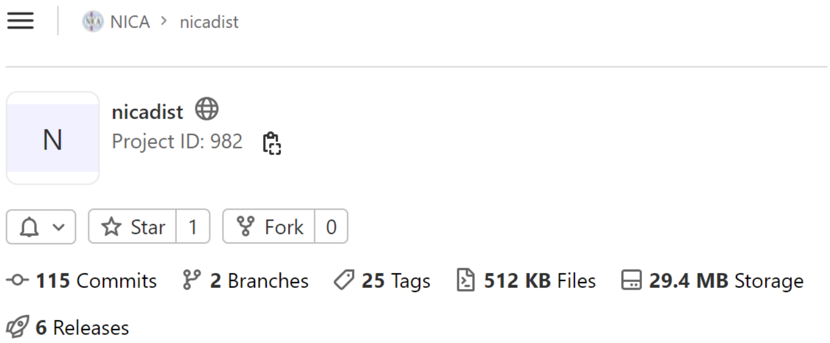


1. Base dependencies (Fair suite, MPDRoot) installation
2. FairSoft clone, build, install, configure
3. FairRoot clone, build, install, configure
4. MPDRoot, clone, build, install, configure

Main Disadvantages

- Base dependencies (>100) different versions, potential source of compatibility issues
- Source build taking many hours for each installation
- Complex procedure with many step-by-step commands, increasing probability of mistake. If error was made usually procedure had to be repeated from scratch

NOW: NICADIST + CVMFS + TOOLBOX



- partial fork of alidist based on aliBuild created to have modular MPDRoot
- builds packages from sources & keeps track of their dependencies
- If any of the build parameters, or package version changes, all dependencies are rebuilt

CVMFS
- Server: stores built modules from nicadist
- Client: auto-installed on local machine, module loading & caching

TOOLBOX
- provides containerized clone of cluster environment on local PC

Buša J. Jr et al.: Unified Software Development and Analysis Environment for MPD Experiment at NICA Collider, 2022

MPDROOT SETUP: USER PERSPECTIVE

INSTALLATION
https://mpdroot.jinr.ru/running-mpdroot-on-local-machine-using-cvmfs/

Running MPDRoot locally using CVMFS

Questions? Click here

INSTALL CVMFS AND TOOLBOX

(Users and Developers)

Supported OS: Fedora, CentOS, AlmaLinux, Ubuntu 22.04, 20.04, Debian 11, 12, Manjaro 21

NOTE: If your OS is based on any of those, then pass it to nica-init script, for example

./nica-init.sh -d Ubuntu -v 20.04

```
[user@fedora ~]$ wget -N https://git.jinr.ru/nica/nicadist/-/raw/master/scripts/nica-init.sh --no-check-certificate
--2021-12-02 00:00:00--  https://git.jinr.ru/nica/nicadist/-/raw/master/scripts/nica-init.sh
.....
.....
2021-12-02 00:00:02 (87.9 MB/s) - ‘nica-init.sh’ saved [10794/10794]

[user@fedora ~]$ chmod +x nica-init.sh && ./nica-init.sh
Installing toolbox on Fedora 38
[sudo] password for user:
INSTALLATION SUCCESSFUL
[user@fedora ~]$ toolbox enter c7-nica-dev
```

USERS

```
[user@toolbox [c7-nica-dev] ~]$ module add mpdroot
```

DEVELOPERS

```
[user@toolbox [c7-nica-dev] ~]$ module add mpddev
[user@toolbox [c7-nica-dev] ~]$ git clone -b dev --recursive git@git.jinr.ru:nica/mpdroot.git
```

ENVIRONMENT & DEPENDENCIES
- the environment & dependencies for the same mpdroot or mpddev versions is **identical**
- no compatibility issues by definition

RELEASES
- release schedule: every 3 months
- “module add mpdroot” loads latest mpdroot release
- old releases can be loaded using specifier
- every release is coupled to its own dependency tree

```
[slavomir@toolbox [c7-nica-dev] ~]$ module add mpdroot/
mpdroot/latest              mpdroot/v22.06.22-1         mpdroot/v23.03.23-1
mpdroot/latest-release      mpdroot/v22.09.22-1         mpdroot/v23.03.23_vhlle-1
mpdroot/v22.04.22-1         mpdroot/v22.12.22-1         mpdroot/v23.06.23-1
[slavomir@toolbox [c7-nica-dev] ~]$ module add mpdroot/v22.04.22-1
[slavomir@toolbox [c7-nica-dev] ~]$ module list
Currently Loaded Modulefiles:
  1) BASE/1.0                       15) lzma/v5.2.3-2                    29) generators/v1.0-4
  2) pythia6/428-alice2-3           16) boost/v1.75.0-4                  30) postgresql/REL_14_2-1
  3) GCC-Toolchain/v10.2.0-alice2-2 17) HepMC/HEPMC_02_06_10-3           31) fmt/8.1.1-1
  4) AliEn-Runtime/v2-19-le-2       18) pythia/v8243-alice1a-4           32) protobuf/v3.15.8-3
  5) FreeType/v2.10.1-4             19) GSL/v1.16-2                      33) eigen3/3.4.0-2
  6) GEANT4/v11.0.1-alice1-1        20) libxml2/v2.9.3-2                 34) asio/v1.19.1-3
  7) lhapdf/v6.2.1-alice2-4         21) XRootD/v5.4.2-alice1-1           35) asiofi/v0.5.1-3
  8) zlib/v1.2.8-2                  22) ROOT/v6-24-06-1                  36) FairLogger/v1.11.0-1
  9) libpng/v1.6.34-3               23) VMC/v2-0-1                       37) ZeroMQ/v4.3.3-3
 10) sqlite/v3.15.0-3               24) vgm/v5-0-1                       38) FairMQ/v1.4.50-1
 11) libffi/v3.2.1-3                25) GEANT4_VMC/v6-1-1                39) FairRoot/v18.6.8-1
 12) Python/v3.6.10-4               26) GEANT3/v4-1-1                    40) mpdroot/v22.04.22-1
 13) OpenSSL/v1.1.1m-1              27) simulation/v1.0-2
 14) Python-modules/1.0-4           28) ofi/v1.14.0-1
```

DESIGN BY CONTRACT

Software Development Stages

Requirements | Architecture / Design | Construction | Testing | **Integration**

INTEGRATION

- Rarely mentioned and almost never planned for
- Reality: multiple independent streams of development
- Assumption: once everyone finishes it will all somehow fit in and work
- Common result: turns out to be a major issue and a significant risk factor of project failure/delay
- Last resort fixes: redesign at late project stages, writing of unnecessary modules

SOLUTION

From the very beginning do:

- Have interfaces
- Agree on interfaces
- Manage interfaces

- Interface control document

All realizations must implement interfaces that are agreed upon

Ensures software modularity, compactness and TESTABILITY

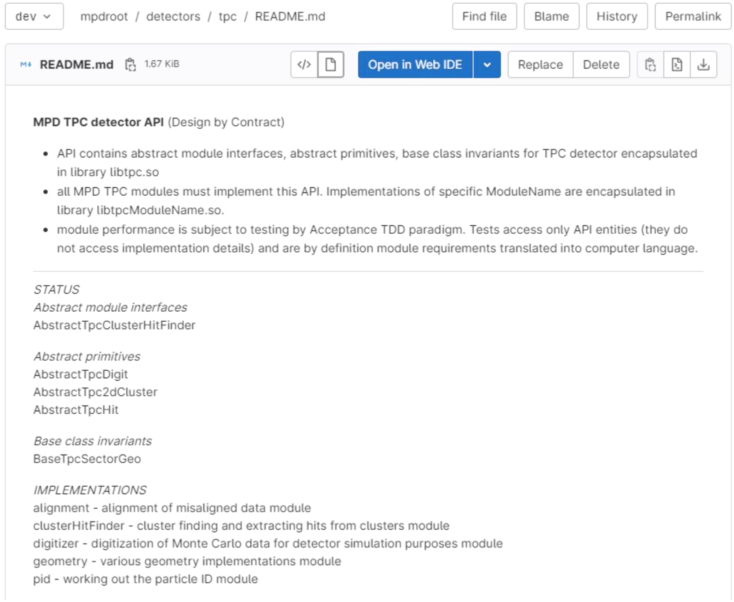
TPC API

API – set of signatures that are exported and available to the users of a library or framework to write their applications.

Key API design notes

- Lead to readable code
- Easy to learn and memorize
- Be complete & stable for proper development and maintenance (be model based)
- Outlast its implementations (invariants)
- Be hard to misuse
- Be easy to extend
- Lead to backward compatibility

Source: *SWEBOK (Software Engineering Body of Knowledge), 2015*



dev mpdroot / detectors / tpc / README.md Find file Blame History Permalink

README.md 1.67 KiB Open in Web IDE Replace Delete

MPD TPC detector API (Design by Contract)

- API contains abstract module interfaces, abstract primitives, base class invariants for TPC detector encapsulated in library libtpc.so
- all MPD TPC modules must implement this API. Implementations of specific ModuleName are encapsulated in library libtpcModuleName.so.
- module performance is subject to testing by Acceptance TDD paradigm. Tests access only API entities (they do not access implementation details) and are by definition module requirements translated into computer language.

STATUS
Abstract module interfaces
AbstractTpcClusterHitFinder

Abstract primitives
AbstractTpcDigit
AbstractTpc2dCluster
AbstractTpcHit

Base class invariants
BaseTpcSectorGeo

IMPLEMENTATIONS
alignment - alignment of misaligned data module
clusterHitFinder - cluster finding and extracting hits from clusters module
digitizer - digitization of Monte Carlo data for detector simulation purposes module
geometry - various geometry implementations module
pid - working out the particle ID module

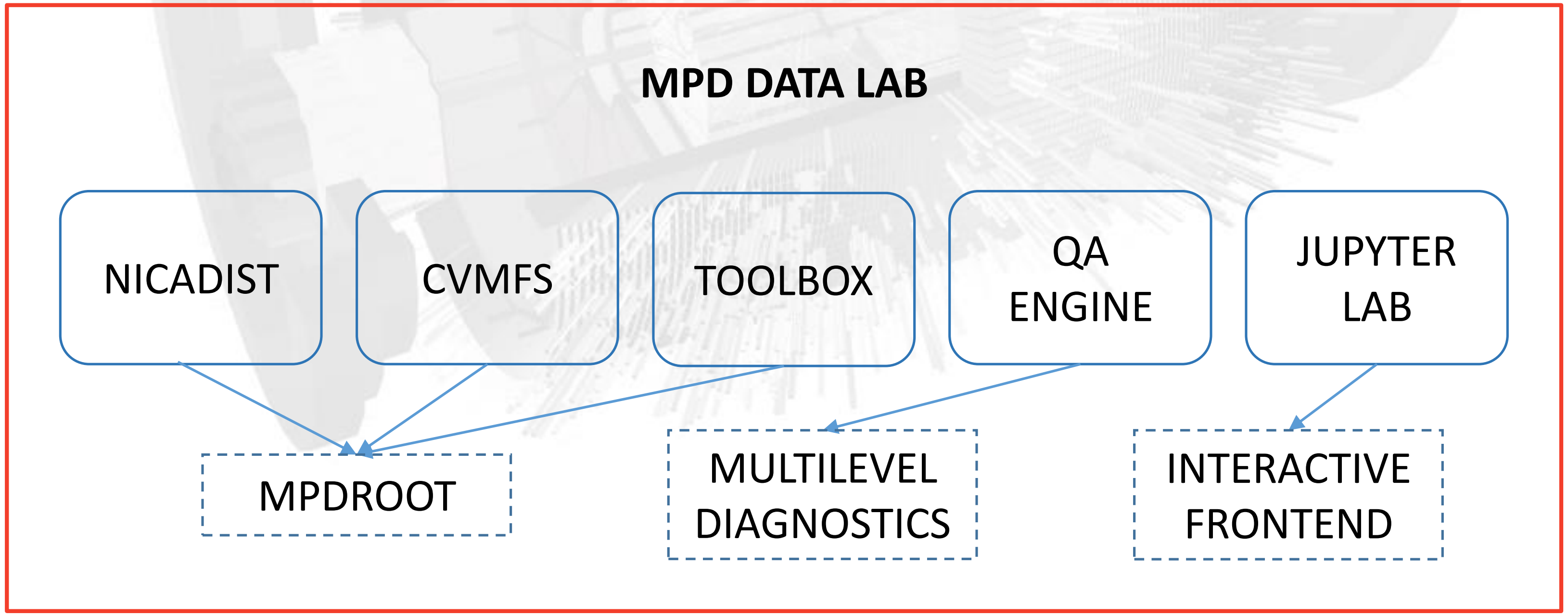
List of the most important things done on MPDRoot

- Complexity reduction
  - downscaling/separation:
    build system, reconstruction/simulation engine, physics
  - codebase cleanup
- Code quality
  - code reviews, code influx control, formatting
  - interfaces, API
  - requirements modeling, acceptance TDD (in progress)
- Build redesign/unified environment
- Stable release schedule
- Support & Maintenance
  - service desk, website, telegram support chat

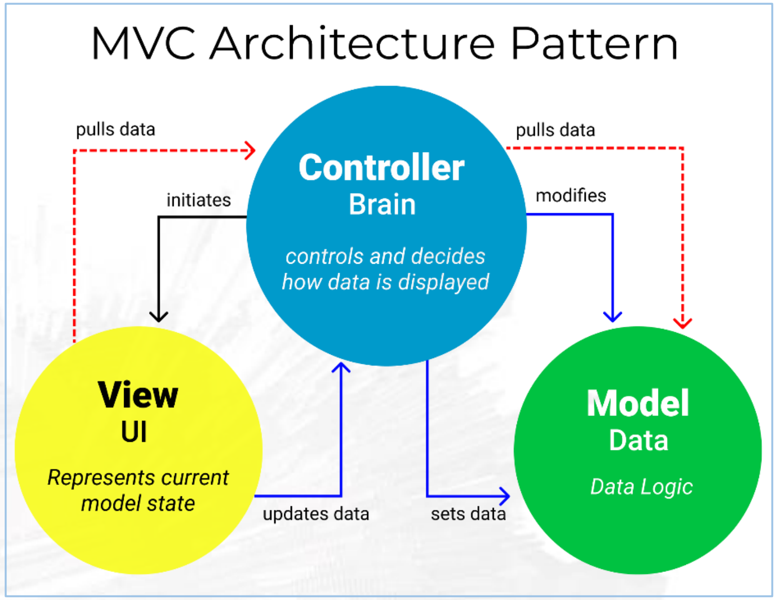
OFFLINE SOFTWARE

The need to have modern data analysis tool

- development **potential** (the variety of possibilities to innovate) directly depends on the properties of development environment
- integrating/modifying the best of latest technologies for the needs of MPD experiment
- clarity, user friendliness, ability to learn on-the-fly



QA ENGINE



QA ENGINE PROPERTIES

- pluggable/switchable reconstruction modules
- QA modes to choose Diagnostics depth
- writing output in terms of MPD primitives into multiple structured root files for modular diagnostics and postprocessing

RUNRECO.C
(v23.09.23 release)

Options:
tpcClustering = ETpcClustering::MLEM
= ETpcClustering::FAST
= ETpcClustering::WAVELET (soon)

qaSetting = EQAMode::OFF
= EQAMode::BASIC
= EQAMode::TPCCLUSTERHITFINDER
= EQAMode::TRACKER (soon)

Upcoming:
tracker = ETracking::DEFAULT
= ETracking::ACTS

Output example: BaseQA_Fast.root, QA_TpcClusterHitFinder_Fast.root
Settings: EQAMode::TPCCLUSTERHITFINDER, ETpcClustering::FAST

REQUIREMENTS: ACCEPTANCE TDD

QA / ACCEPTANCE TDD PARADIGM

- QA overall functional: tools for the analysis, diagnostics & improvement of the process of reconstruction
- critical for overall project success
- QA plots = requirements written in precise test case language

COMPARISON BENCHMARK

- Complex systems: many unknown factors/variables/nonlinearities
- truth best uncovered by comparison of quality properties of the objects of the ***same type*** (standard types defined in interfaces)

QA / ATDD ENVIRONMENT

- Jupyter-Lab with JSRoot
- Custom code injection
- Cell structure with reprocess option
- Graphical output customized on demand
- Algo tuning to real experiment data

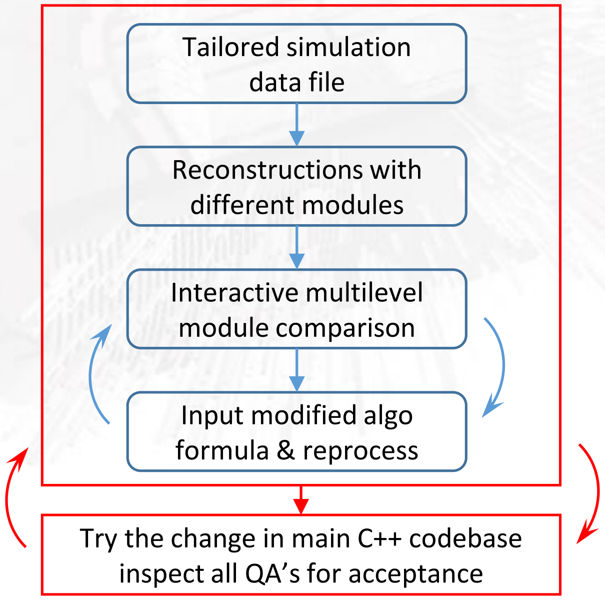
CLUSTERHITFINDER COMPARISON

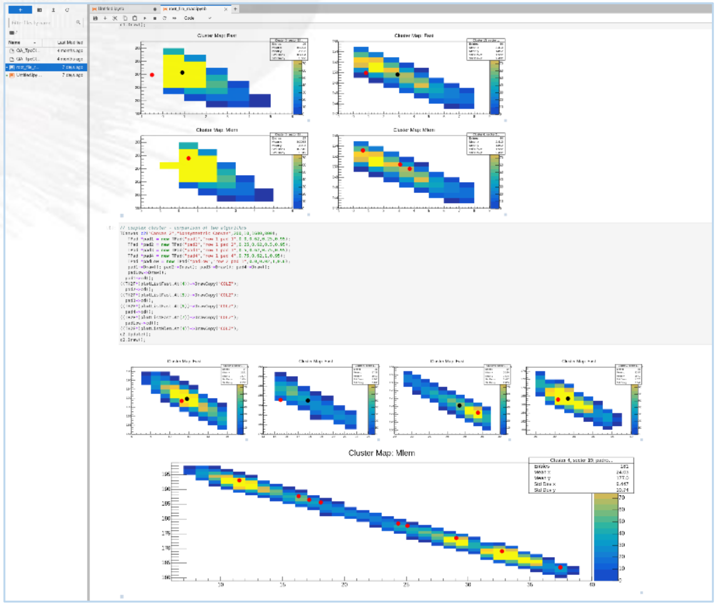
- Mlem
- Fast

ABSTRACTION LEVELS

- Top ....bench.........Reconstruction
- Middle.....component....ClusterHitFinder
- Bottom ....units..........Clustering, Topology, Hit extraction

Interactive workflow example



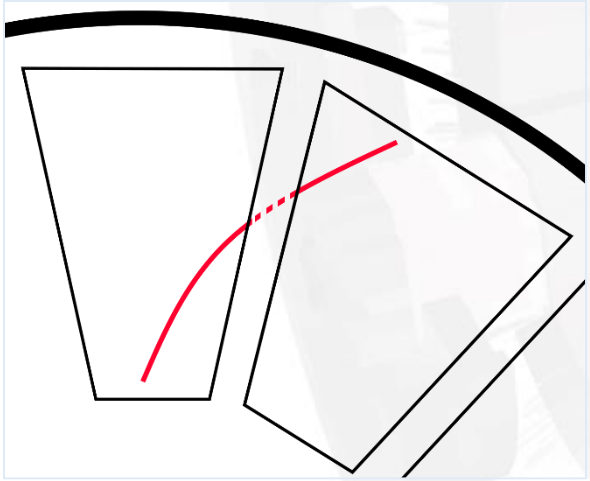


DIAGNOSTICS & RAPID DEVELOPMENT

EXAMPLE: DISCONNECTED TRACKS RETRIEVAL

MC trackID → TPC tracks

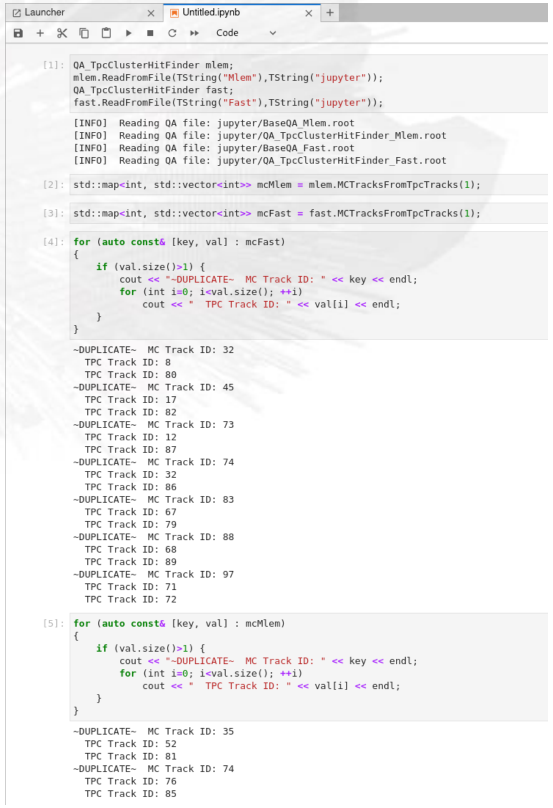
map <int, vector<int>> MCTracksFromTpcTracks(int event);



- to be then used to write, test and evaluate algorithm connecting disconnected tracks
- because of the considerable technical simplification, this work can be outsourced to juniors

RAPID DEVELOPMENT

- Prototyping method – 15 minutes
- Integrating properly into main codebase – half a day !



THE BIG PICTURE

NICADIST
- separate build system
- dependencies handling

CVMFS
- software distribution

TOOLBOX
- Unified environment

Project Management & Support/User Interaction

GITLAB
- codebase
- CI
- testing

SUPPORT
- helpdesk
- telegram channel

WEBSITE
- howtos
- docs
- general info

MPDRoot

ANALYSIS

SIMULATION

RECONSTRUCTION

Mass Production

PWG REQUESTS HANDLING

DIRAC INTERWARE

Computing Infrastructure (MICC & friends)

- supercomputer
- clusters
- storage systems

MPD DATA LAB

TDD ENVIRONMENT
- jupyter-lab
- jsroot
- container

QA
- engine
- gallery

MPD assembly

TPC installation: Oct/Nov 2024

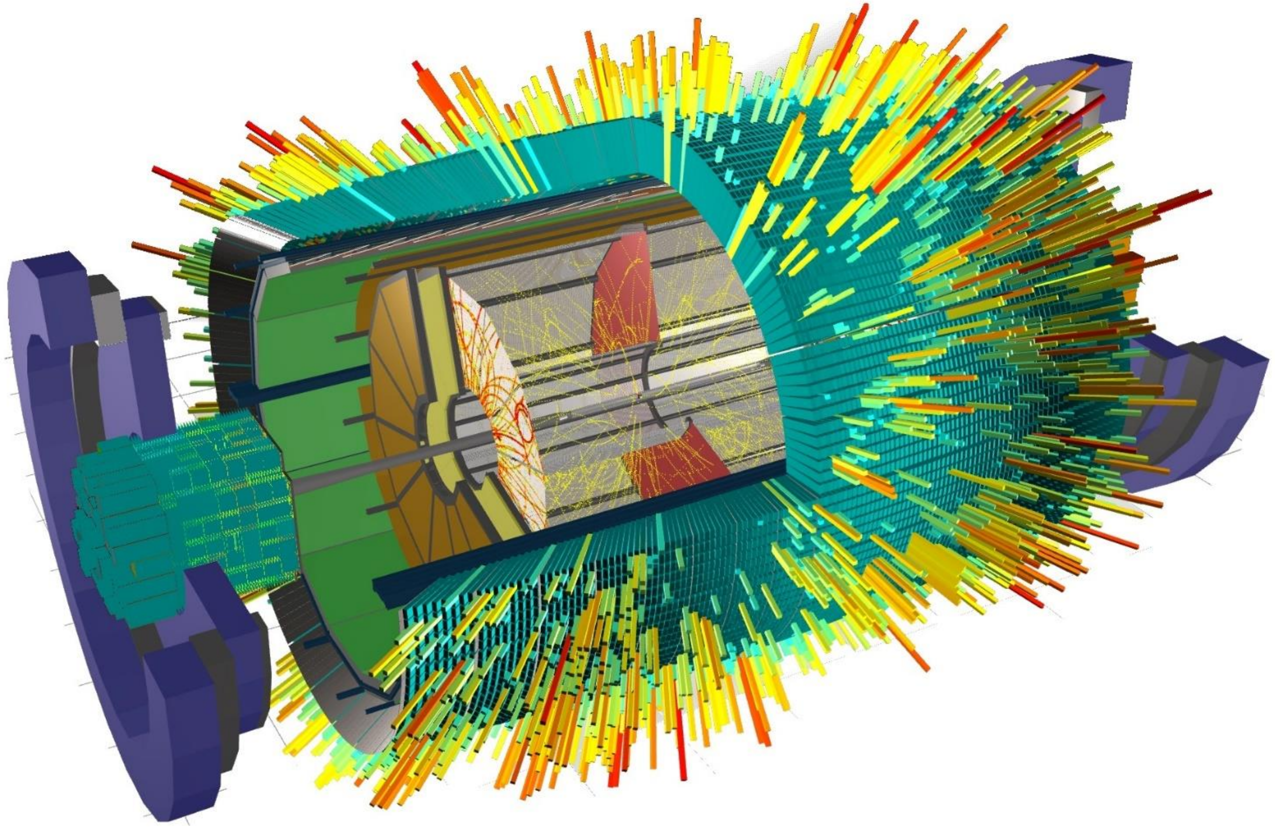
Commissioning: Jan/Feb 2025

ONLINE EVENT DISPLAY
- experiment visualization
- slow control

DATA STORAGE & RETRIEVAL

DETECTOR CALIBRATION
- alignment
- noise level
- digitalization delay

Thank You !



MPD Software Development & Computing Team

Rogachevsky O.	Coordinator
Krylov V., Krylov A.	Online MPD Event Display
Moshkin A., Pelevanyuk I.	Mass Production
Bychkov A.	Detector Simulation
Kuzmin V.	Detector Alignment
Podgainy D., Zuev M.	Supercomputing
Alexandrov E., Alexandrov I.	Databases
Balashov N.	Gitlab Support
Belyakov D.	Network Infrastructure
Belecky P., Kamkin A.	Acts Tracker
Busa J.	Build System
Hnatic S.	Architecture