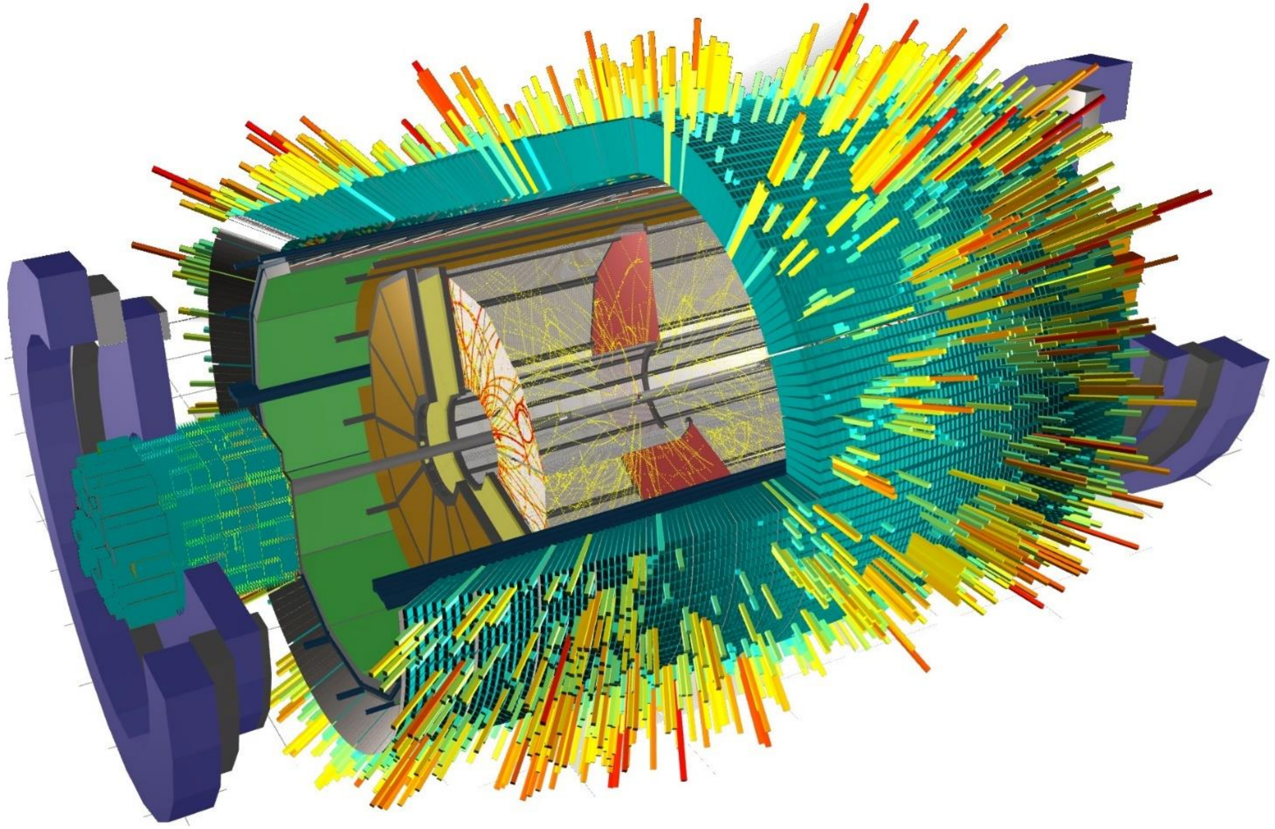
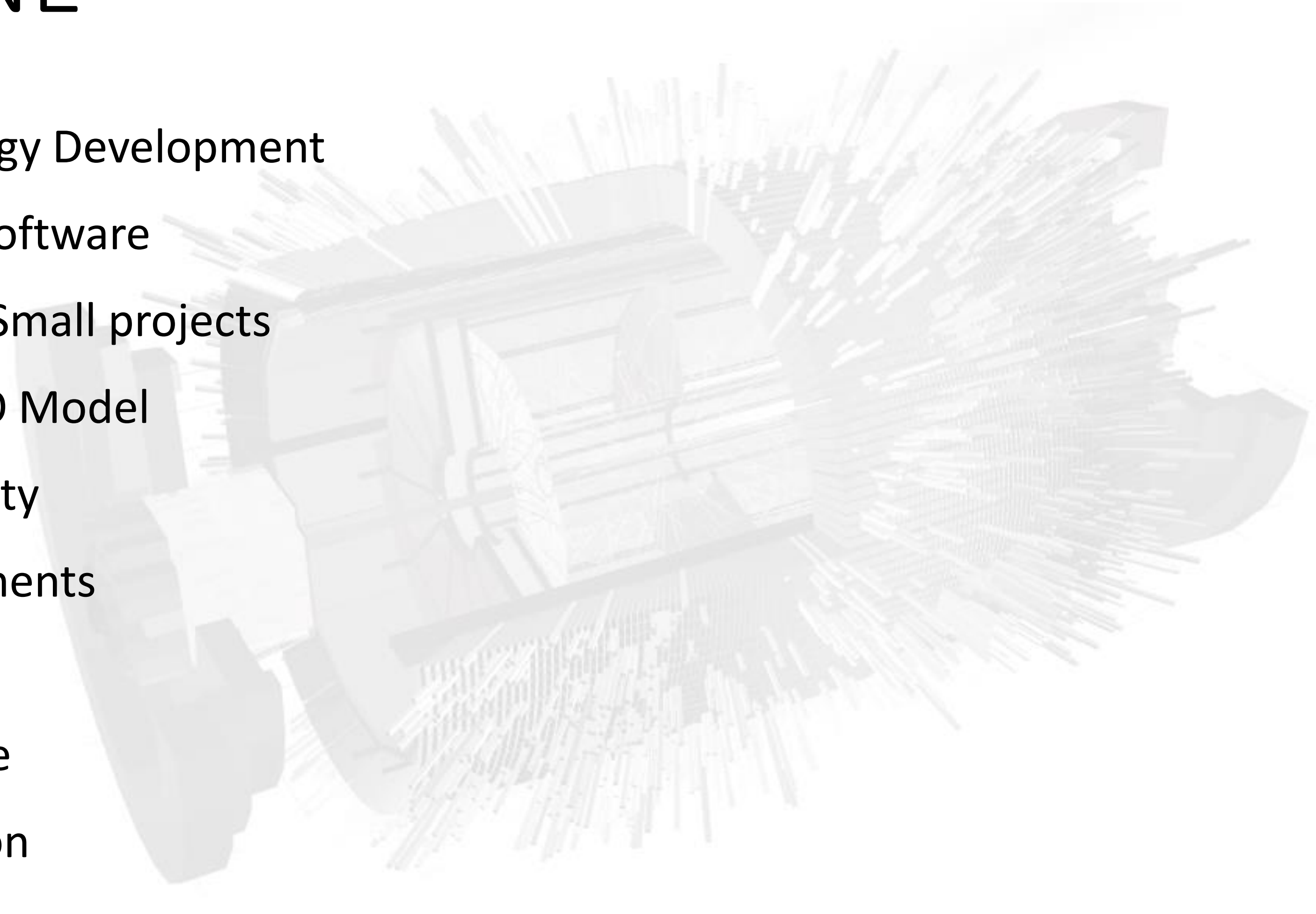
MPDRoot
from R&D towards Software

HNATIC Slavomir



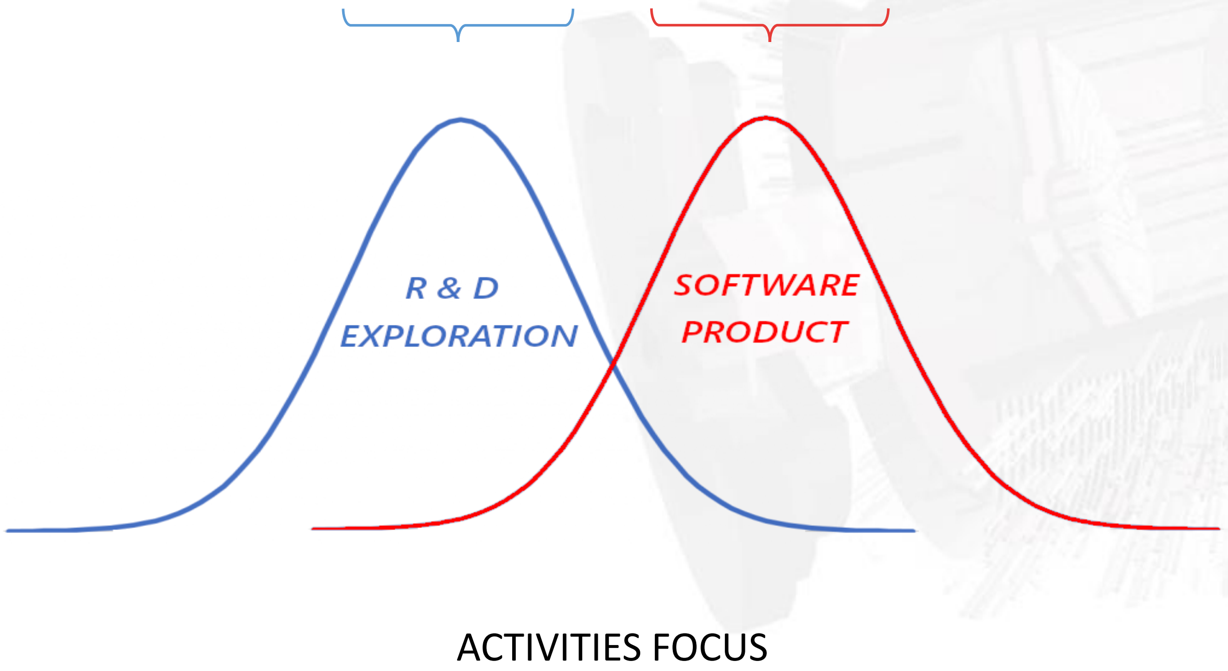
OUTLINE

- Technology Development
- R&D vs Software
- Large vs Small projects
- COCOMO Model
- Complexity
- Requirements
- Coding
- Codebase
- Innovation



TECHNOLOGY DEVELOPMENT

Technology Development =
Scientific Theory + Engineering Practice + Economy



"…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)

R&D vs SOFTWARE ENGINEERING

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

LARGE PROJECTS vs SMALL PROJECTS

WHY IS THIS IMPORTANT ?

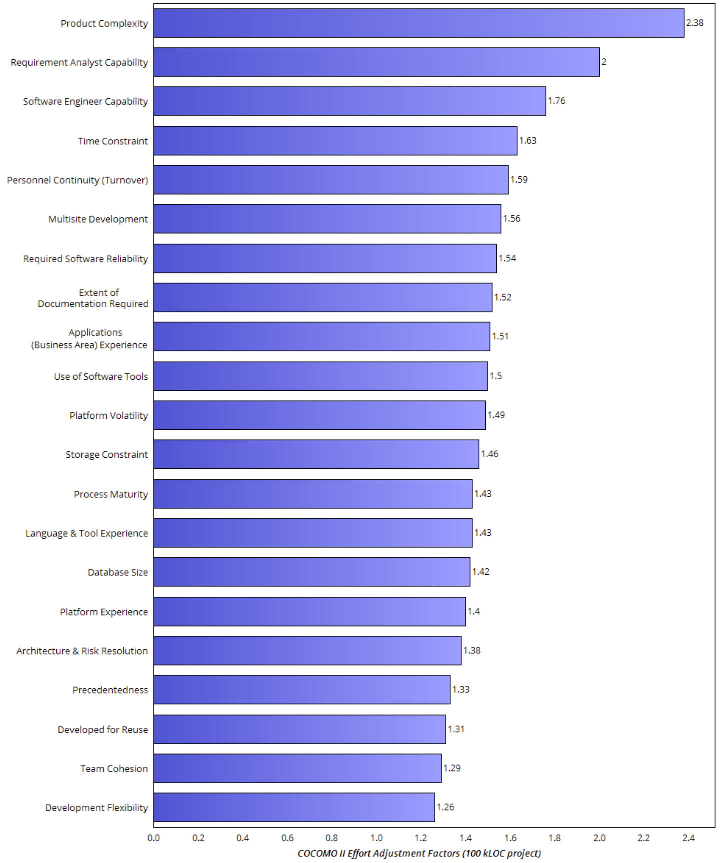
- MPD – large project by duration, computational size, data volumes, projected user number
- Large codebase with continuous substantial influx of inputs expected
- Small project success (single R&D concept) does not prepare for large project succes
- Software = foundation for Research / R&D
- Software stability, quality, efficiency – success critical factor
- Change of focus, build of additional skill sets critical as projects become larger
- Large project core influences:

  - size (scaling)
  - defects handling
  - dealing with uncertainty
  - human variation
  - synergy

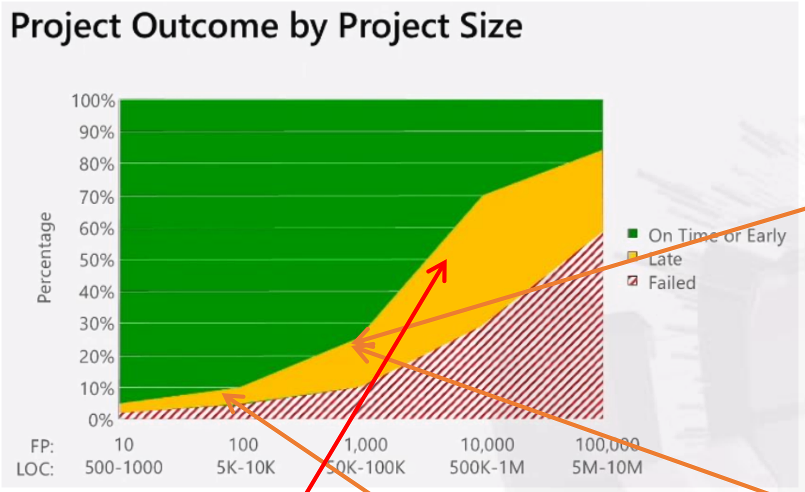
SOFTWARE PROJECT DYNAMICS

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

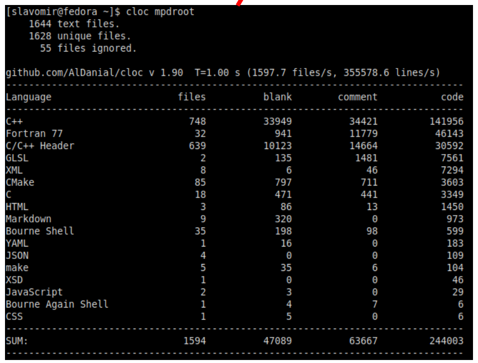
- Most rigorous statistical analysis of software projects using data from historic projects
- Results expressed in “effort adjustment factors”, these describe software project dynamics, used to gain insight to adjust the development strategy
- Requirements Analyst Capability factor 2 means project with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



COMPLEXITY



Applied Software Measurement, C. Jones (2008)

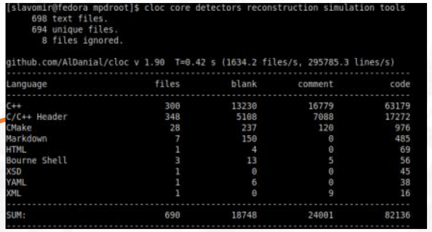


```
[slavomir@fedora ~]$ cloc mpdroot
    1644 text files.
    1628 unique files.
      55 files ignored.

github.com/AlDanial/cloc v 1.90  T=1.00 s (1597.7 files/s, 355578.6 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                            748          33949          34421         141956
Fortran 77                      32            941          11779          46143
C/C++ Header                   639          10123          14664          30592
GLSL                             2            135           1481           7561
XML                              8              6             46           7294
CMake                           85            797            711           3603
C                               18            471            441           3349
HTML                             3             86             13           1450
Markdown                         9            320              0            973
Bourne Shell                    35            198             98            599
YAML                             1             16              0            183
JSON                             4              0              0            109
make                             5             35              6            104
XSD                              1              0              0             46
JavaScript                       2              3              0             29
Bourne Again Shell               1              4              7              6
CSS                              1              5              0              6
-------------------------------------------------------------------------------
SUM:                          1594          47089          63667         244003
-------------------------------------------------------------------------------
```

April 2021

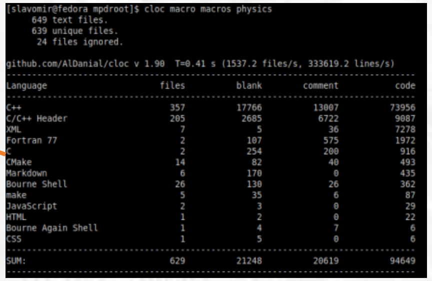
Backend



```
[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
-------------------------------------------------------------------------------
```

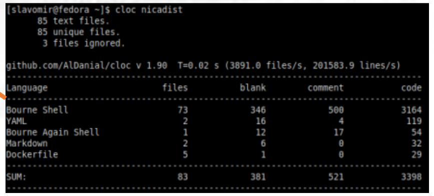
Physics



```
[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
-------------------------------------------------------------------------------
```

Build



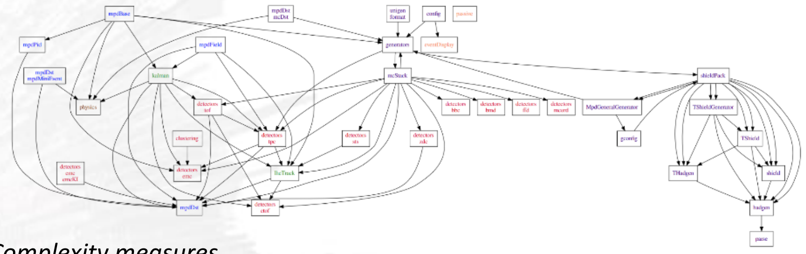
```
[slavomir@fedora ~]$ cloc nicadist
      85 text files.
      85 unique files.
       3 files ignored.

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
-------------------------------------------------------------------------------
```

April 2022

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

Codebase Restructuring & Cleanup



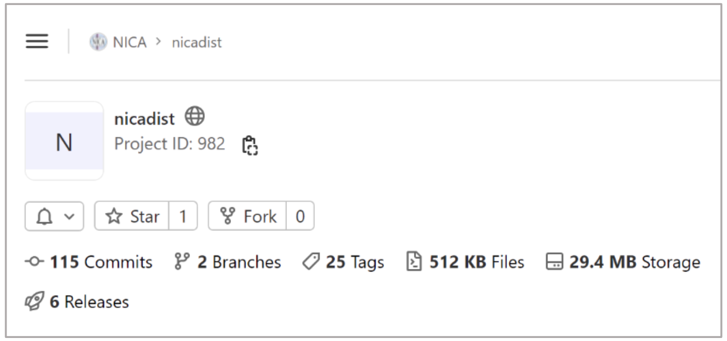
Complexity measures

Lines Of Code - crude (easy to obtain)

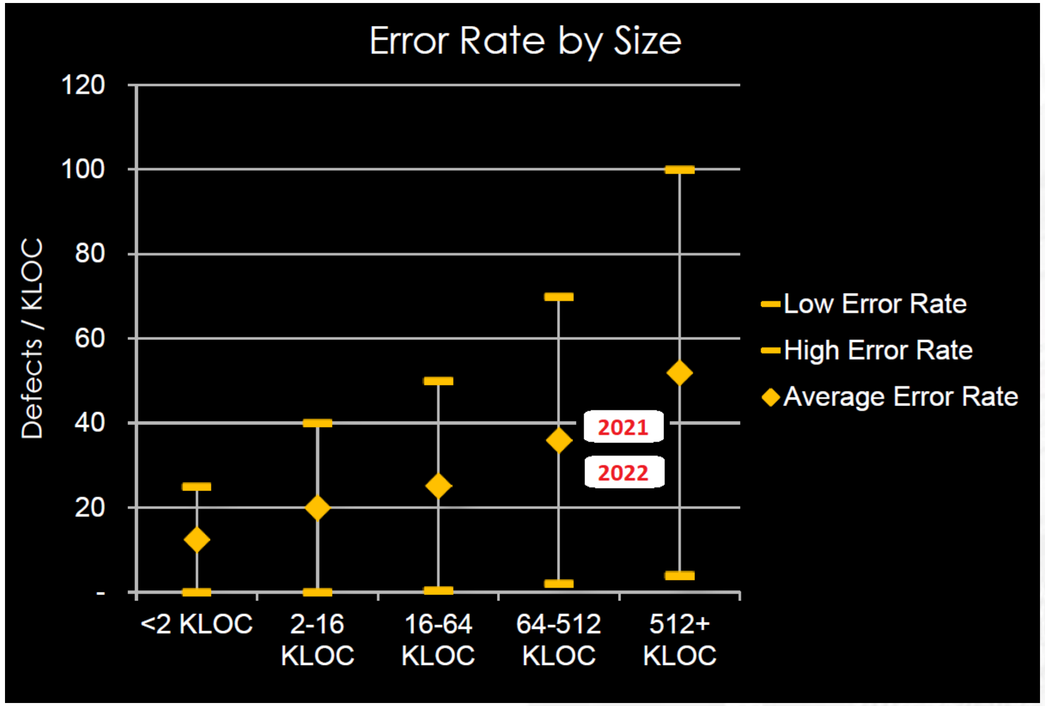
Functional Points - exact (difficult)

- structure dependent: tight vs loose coupling

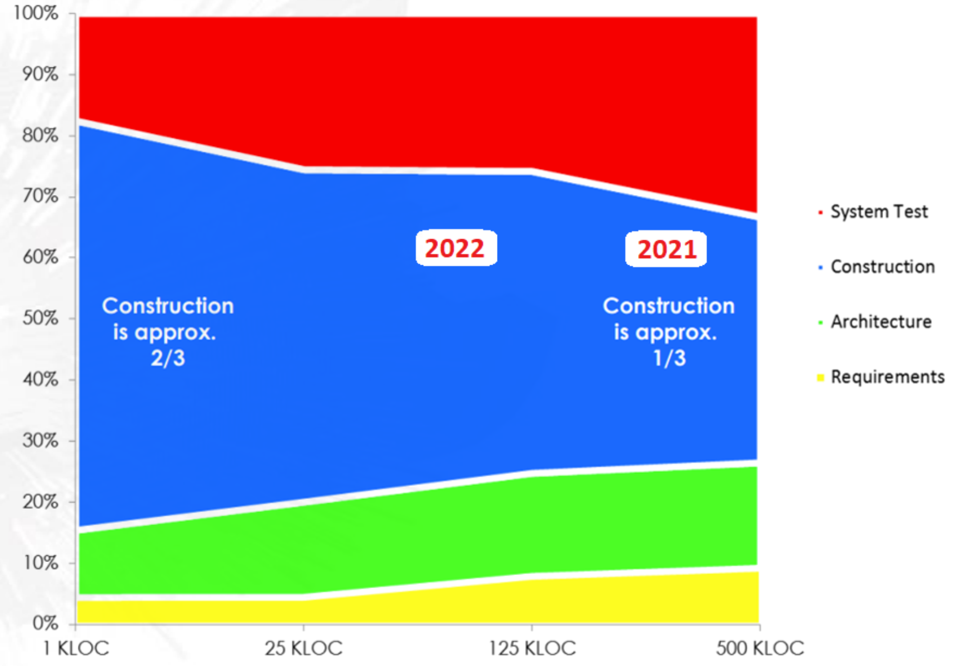
Build System Separation



DOWNSCALING EFFECT



Project Activity Mix by Size

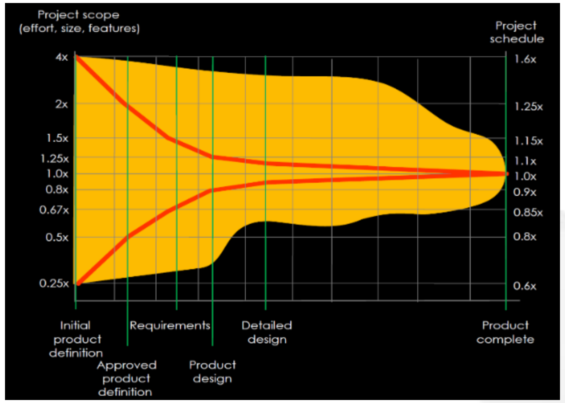


How Not to be Surprised in Software Development,
S. McConnell (2012)

COMPACTNESS & MODULARITY = Long term **critical** objective !!!

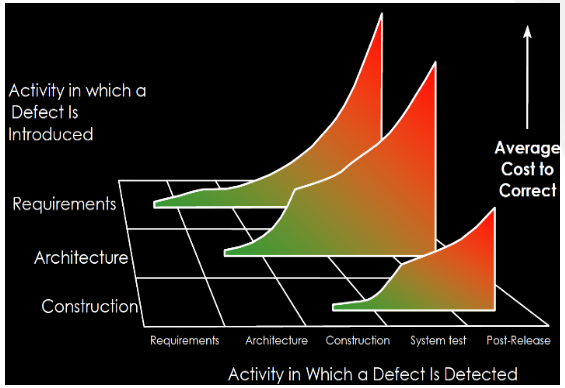
REQUIREMENTS MODELING

Uncertainty: Cloud vs Cone



- variability/convergence of the project to desired result
- the cone does not narrow by itself
- target sources of uncertainty early

Defects handling



- the later the defect is fixed, the more it costs to correct
- try to detect defects early
- do not cumulate technical debt – fix defects asap

MPDROOT

- High level product spec

  - in process, once ready
    subject for approval

- Requirements modeling

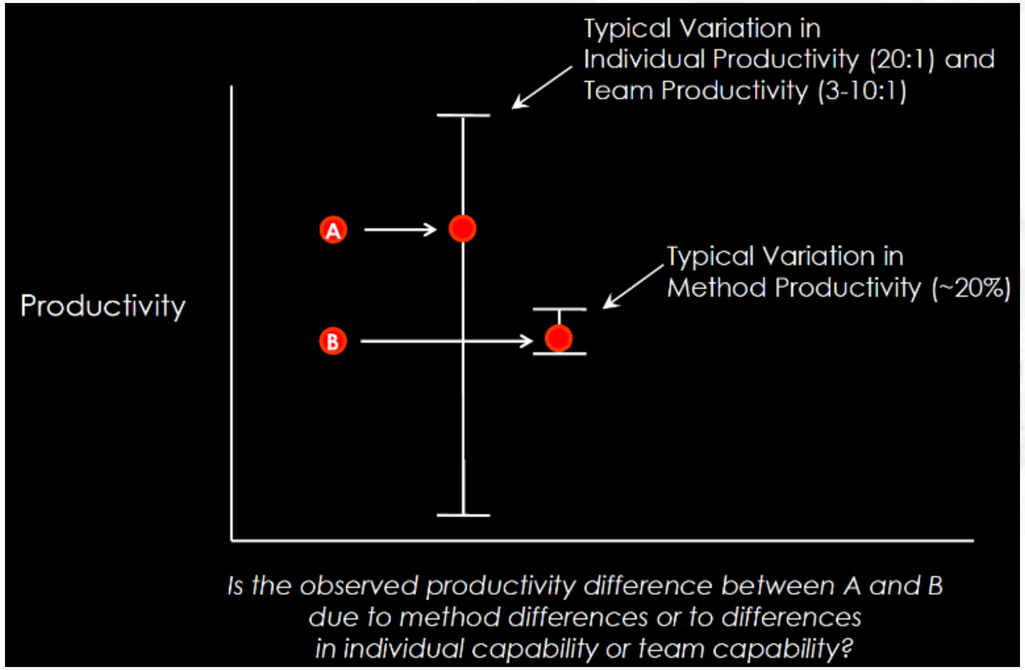
  functional
  - user personas
  - user stories
  - use cases

  non-functional
  - system performance

CODING

Variation: Human vs Development Process

- By far: Capability > Process



Is the observed productivity difference between A and B due to method differences or to differences in individual capability or team capability?

MPDROOT CODING RULES

Basic truths

1. It’s harder to read the code, than to write it
2. Capability based approach being the most effective

Focus

- readability
- design
- general rules:

https://mpdroot.jinr.ru/mpdroot-naming-convention/

CODEBASE

MPDROOT

NO OBJECT ORIENTED DESIGN

TECHNOLOGY DEVELOPMENT METHODOLOGIES CANNOT BE USED EFFECTIVELY

- No interfaces **at all**
- No **abstraction hierarchy**
- Procedural code written using C++ syntax ridden with **OO design antipatterns**

DO:

- Object Oriented **Analysis**
- Design **Interfaces**
- Design **Class Invariants**
- Remove antipatterns **(global state, God class)**
- Use **OO design patterns**

FAIRROOT

OBJECT ORIENTED DESIGN

(sort of)

Single responsibility principle

Open/Closed principle

Liskov Substitution principle

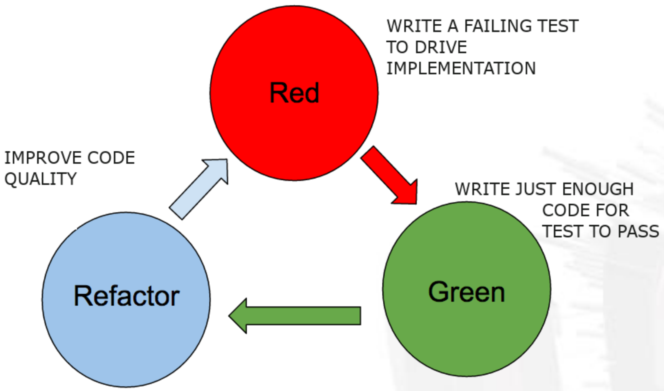
Interface Segregation principle

Specific Interfaces

Dependency Inversion principle

Software Entities designed in direction from high to low abstraction

TDD: MPDROOT


WRITE A FAILING TEST TO DRIVE IMPLEMENTATION
Red
IMPROVE CODE QUALITY
WRITE JUST ENOUGH CODE FOR TEST TO PASS
Refactor
Green

EFFICIENT DEVELOPMENT CONSISTS OF

1. Define module external behavior
2. Develop working prototype
3. Refactor

- precision (output accuracy improvement)
- performance (structural improvement)

DESIGNING TESTS ON MULTIPLE ABSTRACTION LEVELS

Test level hierarchy “system / component / unit“ adapted for MPDRoot’s backend:

- Top level…………system (bench) tests…….QA
- Middle level……….component tests……….specific reconstruction FairTasks (invariant interfaces)
- Bottom level……………unit tests……………interface units (invariant pure virtual methods)

TDD: PILOT USE CASE

Cluster Hit Finder

Preparatory work

- get rid of geometry singleton
- create **invariant** Base class for geometry

Interface

- inheriting from FairTask
- test-driven design
- dependencies passed by **injection**
- clusterhitfinder units: findClusters, findHits

Implementation

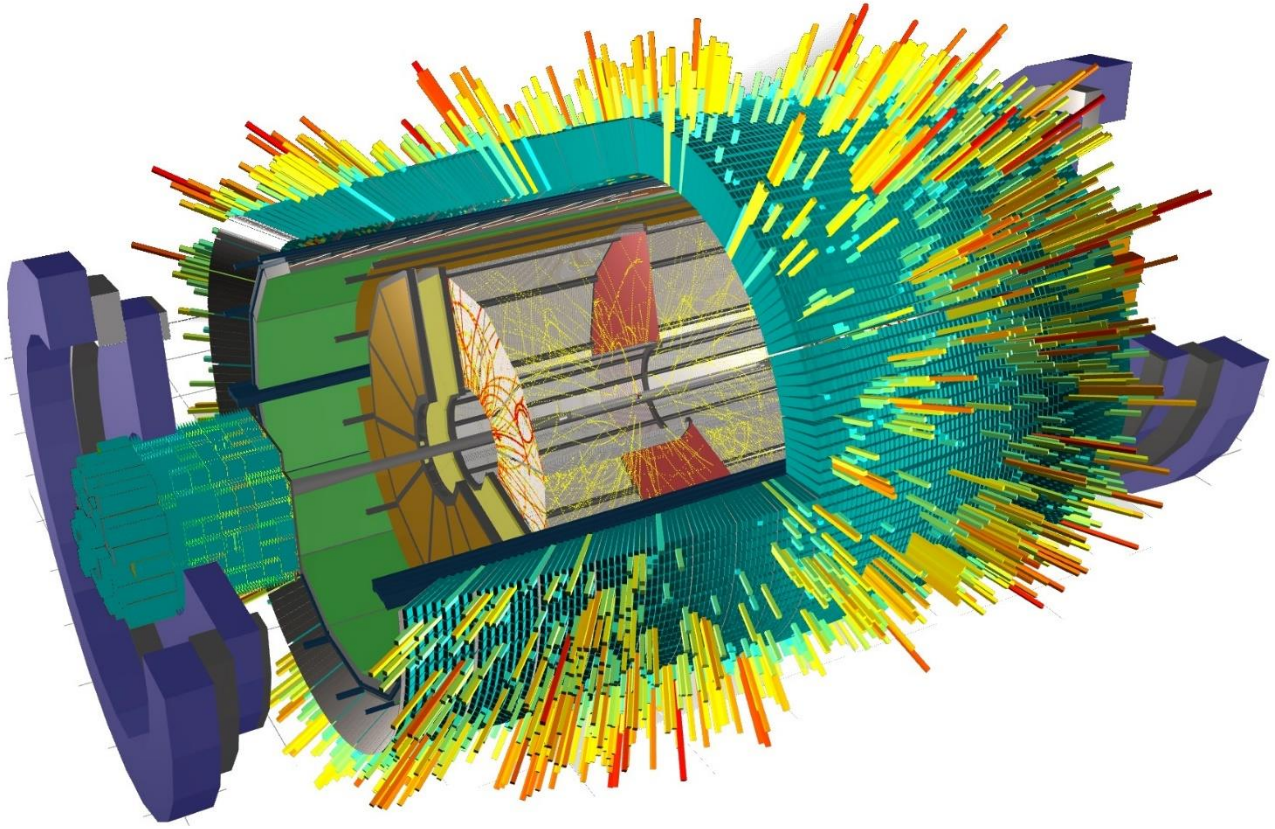
- current Mlem algorithm to be adapted to interface (criterion: reconstruction identity)
- new fast clusterhitfinder to be adapted to interface
- both algorithms are standardized and testable on levels of:
  - pure virtual methods
  - interface
  - reconstruction

TDD

- standardized criteria
  - precision
  - performance
- multilevel analysis
  - improvement of which part has the most significant effect?
- hybrid algorithms
- long term effective strategy
- data-driven tests varied depending on improvement requirements

Thank You !

Q & A

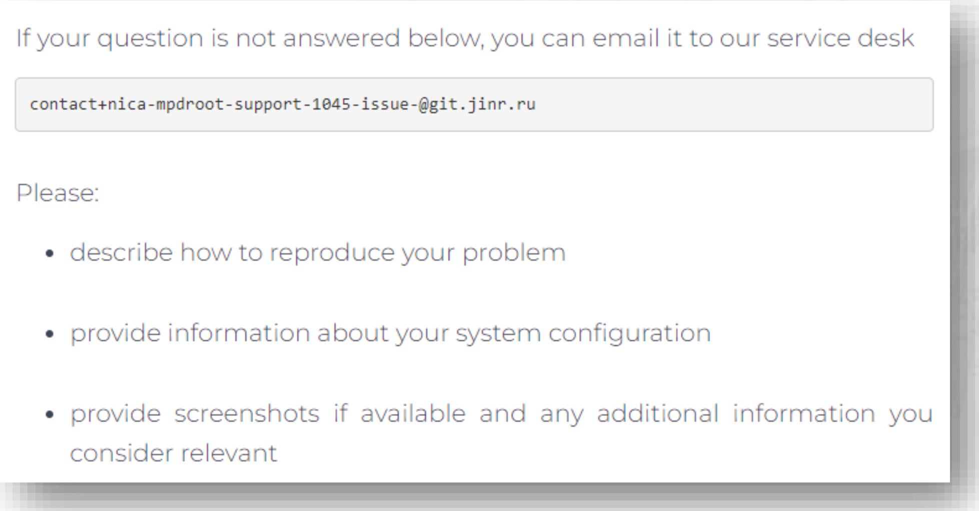


USERS

“User Involvement – **critical** project success factor”
CHAOS Report 2015, Standish Group

SERVICE DESK for Questions

http://mpdroot.jinr.ru/q-a/

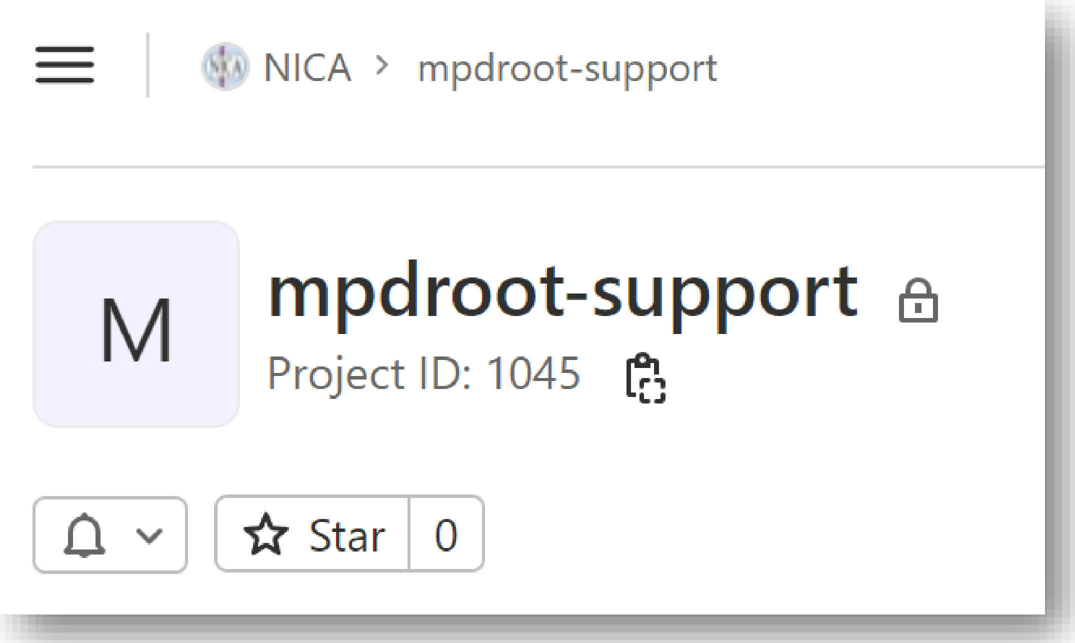


If your question is not answered below, you can email it to our service desk

```
contact+nica-mpdroot-support-1045-issue-@git.jinr.ru
```

Please:

- describe how to reproduce your problem
- provide information about your system configuration
- provide screenshots if available and any additional information you consider relevant



mpdroot-support

Project ID: 1045