AliRoot: Unit and regression testing

Outlook

Unit testing

Regression testing

*For other software tests see next presentation: Release QA cluster on CERN Agile structure

User stories

Proposal

Unit testing

Unit testing (component testing): verify the functionality of a specific section of code

- Ensure that the building blocks of the software work independently from each other
- In an object-oriented environment usually at the class level, but could be used for individual methods
- Unit testing aims to eliminate construction errors before code is promoted to QA

Benefits:

- Finds problems early
- Facilitates change (regression testing)
- Simplifies integration
- Documentation (basic understanding of the unit's interface)

Regression testing

Uncover new software bugs or regressions after software changes

- enhancements, patches or configuration changes

The intent of regression testing: no new bugs in old code

No recurrence of old bugs

Common methods of regression testing:

- rerunning previously completed tests
- checking whether program behavior has changed
- checking whether previously fixed faults have re-emerged

Efficiency

$\varepsilon_i < \varepsilon_{i+1} - n^* \sigma\varepsilon_{i+1}$

Resolution

$\sigma_i < \sigma_{i+1} - n^* \sigma_{\sigma i+1}$

Reconstruction. bias

$|\mu_{i"}| < |\mu_{i+1}|$

Execution time

$T_i < T_{i+1} - n^* \sigma_{Ti+1}$

Memory

$VM_i < VM_{i+1} - n^* \sigma_{VMi+1}$

User story

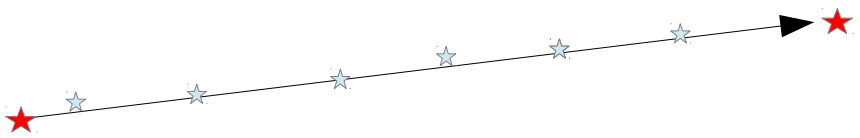
User story - Test of Kalman Filter (0)

As a tracking developer and user, I want to automatically test basic components of the tracking code, to ensure their reliability

- Propagation in the B field
  - 3D (OFFLINE) resp. 1D (HLT) propagation
  - Precision, bias vs timing
- Propagation and energy loss correction and multiple scattering
  - Step size/precision/time
- Kalman update
  - Consistency of error estimates

Point A: e.g. TPC

Point B: e.g. TRD



What step size we need to get given precision?
How many updates do we need to …
How much in performance we pay for simplifications ?

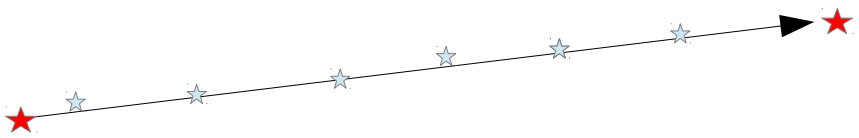
User story - Test of Kalman Filter (1)

Oracle to determine whether a test has passed or failed:

- a consistency oracle that compares the results of one test execution to another for similarity - Tracking propagation vs MC propagation
- a statistical oracle that uses statistical characteristics: analytical expectation bias negligible to internal error (due to MS and energy loss fluctuation)
- regression testing to compare with previous versions of the existing software
  - DB support needed

Point A: e.g. TPC

Point B: e.g. TRD



What step size we need to get given precision?
How many updates do we need to …
How much in performance we pay for simplifications

Did performance improve in respect to previous version

User story - Test of AliKF package (0)

Since my code uses vertexing package, I want to automatically test basic components of the AliKF classes, to ensure their reliability.

- Secondary vertex constraint
- Primary vertex constraint
- Mass constraint
- Propagation in B-field, material budget correction

As a user of the vertex package, I want to characterize performance of the package for individual AliRoot releases in predefined place

- Performance for ideal MC input
- Performance for realistic raw input

User story - Test of AliKF package

Oracle to determine whether a test has passed or failed:

- a consistency oracle that compares the results of one test execution to another for similarity (MC true/reconstructed data):
  - Each constraint has to improve momentum resolution
  - First 2 constraints should improve the inv. mass resolution
- a statistical oracle that uses statistical characteristics:
  - inv. mass bias negligible to intrinsic resolution
  - pulls (normalized errors) close to one
- regression testing to compare with previous versions of the existing software

Other user story examples.

As user of the TTreeStream and TTreeSRedirector, I want to automatically test stability, compression factor and speed .

As a user of the TStatTookit, I want automatically test individual fit and statistical function.

- Linear fitting
- Linear fitting with constraints
- LTM statistic
- Robust Gaussian fit

Proposal

Proposal

Individual physics and PWGPP groups define their tests, which will be executed automatically with given time periodicity

- experience from Root, CMS and FairRoot

Tests procedure should be integrated to the AliRoot test framework

- cdash like prototype existing but not running on regular basis

Consideration/Questions

Test should be shell script like

- CTest like not cppTest like

Export status and logs

- … as in Cdash

Additional characteristic variables, plots registered on predefined place for further reference

Trending variables needed for regression test registered and accessed in the DB

Where to keep the test code?

- Inside of classes itself (not recommended)
- Special code macros? How to keep track?
  - Naming convention?

Which DB to use?

- current test in $ALICE_ROOT/test/ uses TMonalisaWriter
- how to retrieve information back?

Test input data

My test code structure (example)

Directory naming convention in $ALICE_ROOT:

- $ALICE_ROOT/test/test<class>
- e.g testAliAnalysisTaskFiltered/testAliAnalysisTaskFiltered/

Executable naming convention:

- AliAnalysisTaskFilteredTest.sh

Trending (regression) variables exported in log files:

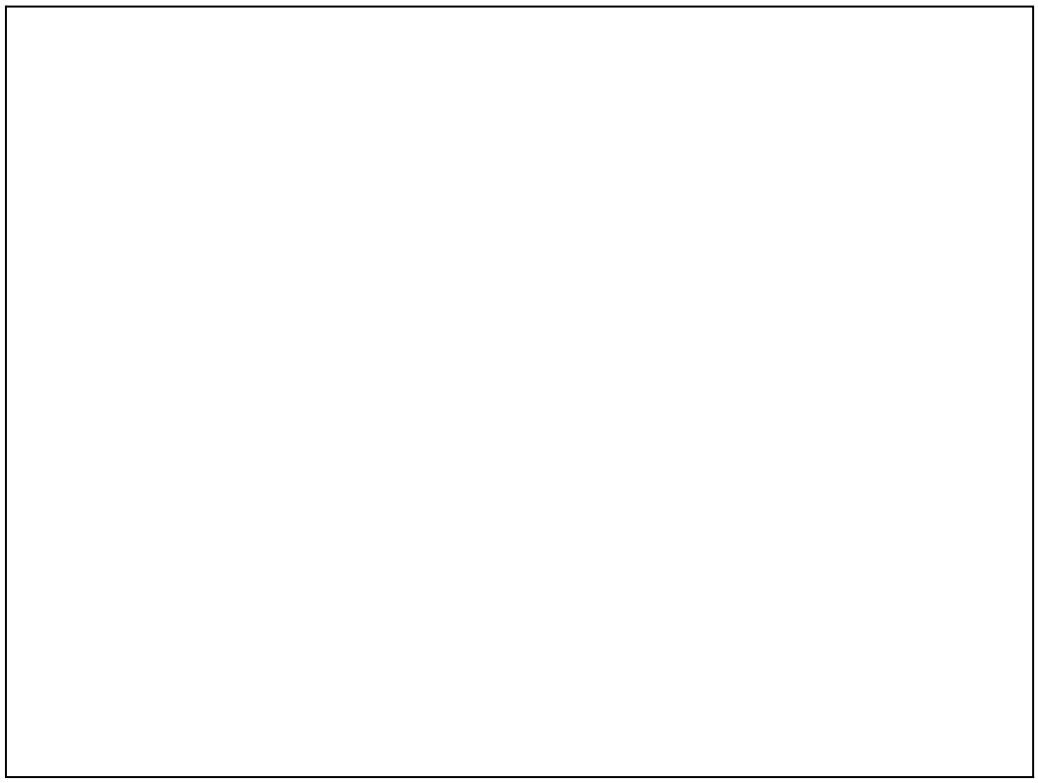
- Parsing using Key ^#UnitTest:
- Test tag name
- Primary and secondary tag name
- Tag value

Test output directory naming convention (in addition to CDash like)

- e.g: http://cern.ch/alitest/<aliroot-tag>/test<class>/

Test configuration file:

- $ALICE_ROOT/test/configuration/
- Description of path to test data, reps. list of input data files
- OCDB setup (prefix)



Outlook

Unit testing

Regression testing

*For other software tests see next presentation: Release QA cluster on CERN Agile structure

User stories

Proposal

Unit testing

Unit testing (component testing): verify the functionality of a specific section of code

- Ensure that the building blocks of the software work independently from each other
- In an object-oriented environment usually at the class level, but could be used for individual methods
- Unit testing aims to eliminate construction errors before code is promoted to QA

Benefits:

- Finds problems early
- Facilitates change (regression testing)
- Simplifies integration
- Documentation (basic understanding of the unit's interface)

Regression testing

Uncover new software bugs or regressions after software changes

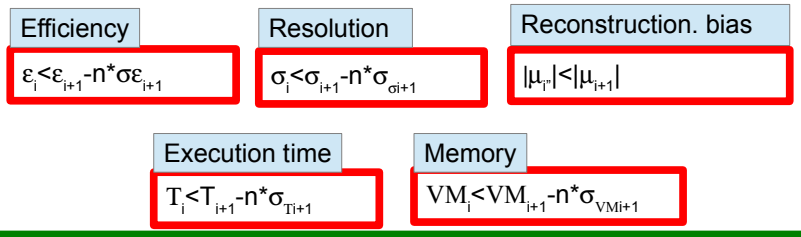
- enhancements, patches or configuration changes

The intent of regression testing: no new bugs in old code

No recurrence of old bugs

Common methods of regression testing:

- rerunning previously completed tests
- checking whether program behavior has changed
- checking whether previously fixed faults have re-emerged



User story

User story - Test of Kalman Filter (0)

As a tracking developer and user, I want to automatically test basic components of the tracking code, to ensure their reliability

- Propagation in the B field
  - 3D (OFFLINE) resp. 1D (HLT) propagation
  - Precision, bias vs timing
- Propagation and energy loss correction and multiple scattering
  - Step size/precision/time
- Kalman update
  - Consistency of error estimates

Point A: e.g. TPC

Point B: e.g. TRD

What step size we need to get given precision?
How many updates do we need to …
How much in performance we pay for simplifications ?

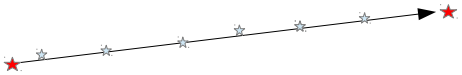
User story - Test of Kalman Filter (1)

Oracle to determine whether a test has passed or failed:

- a consistency oracle that compares the results of one test execution to another for similarity - Tracking propagation vs MC propagation
- a statistical oracle that uses statistical characteristics: analytical expectation bias negligible to internal error (due to MS and energy loss fluctuation)
- regression testing to compare with previous versions of the existing software
  - DB support needed

Point A: e.g. TPC

Point B: e.g. TRD



What step size we need to get given precision?
How many updates do we need to …
How much in performance we pay for simplifications

Did performance improve in respect to previous version

User story - Test of AliKF package (0)

Since my code uses vertexing package, I want to automatically test basic components of the AliKF classes, to ensure their reliability.

- Secondary vertex constraint
- Primary vertex constraint
- Mass constraint
- Propagation in B-field, material budget correction

As a user of the vertex package, I want to characterize performance of the package for individual AliRoot releases in predefined place

- Performance for ideal MC input
- Performance for realistic raw input

User story - Test of AliKF package

Oracle to determine whether a test has passed or failed:

- a consistency oracle that compares the results of one test execution to another for similarity (MC true/reconstructed data):
  - Each constraint has to improve momentum resolution
  - First 2 constraints should improve the inv. mass resolution
- a statistical oracle that uses statistical characteristics:
  - inv. mass bias negligible to intrinsic resolution
  - pulls (normalized errors) close to one
- regression testing to compare with previous versions of the existing software

Other user story examples.

As user of the TTreeStream and TTreeSRedirector, I want to automatically test stability, compression factor and speed .

As a user of the TStatTookit, I want automatically test individual fit and statistical function.

- Linear fitting
- Linear fitting with constraints
- LTM statistic
- Robust Gaussian fit

Proposal

Proposal

Individual physics and PWGPP groups define their tests, which will be executed automatically with given time periodicity

- experience from Root, CMS and FairRoot

Tests procedure should be integrated to the AliRoot test framework

- cdash like prototype existing but not running on regular basis

Consideration/Questions

- Test should be shell script like
  - CTest like not cppTest like
- Export status and logs
  - … as in Cdash
- Additional characteristic variables, plots registered on predefined place for further reference
- Trending variables needed for regression test registered and accessed in the DB
- Where to keep the test code?
  - Inside of classes itself (not recommended)
  - Special code macros? How to keep track?
    - Naming convention?
- Which DB to use?
  - current test in $ALICE_ROOT/test/ uses TMonalisaWriter
  - how to retrieve information back?
- Test input data

My test code structure (example)

Directory naming convention in $ALICE_ROOT:

- $ALICE_ROOT/test/test<class>
- e.g testAliAnalysisTaskFiltered/testAliAnalysisTaskFiltered/

Executable naming convention:

- AliAnalysisTaskFilteredTest.sh

Trending (regression) variables exported in log files:

- Parsing using Key ^#UnitTest:
- Test tag name
- Primary and secondary tag name
- Tag value

Test output directory naming convention (in addition to CDash like)

- e.g: http://cern.ch/alitest/<aliroot-tag>/test<class>/

Test configuration file:

- $ALICE_ROOT/test/configuration/
- Description of path to test data, reps. list of input data files
- OCDB setup (prefix)