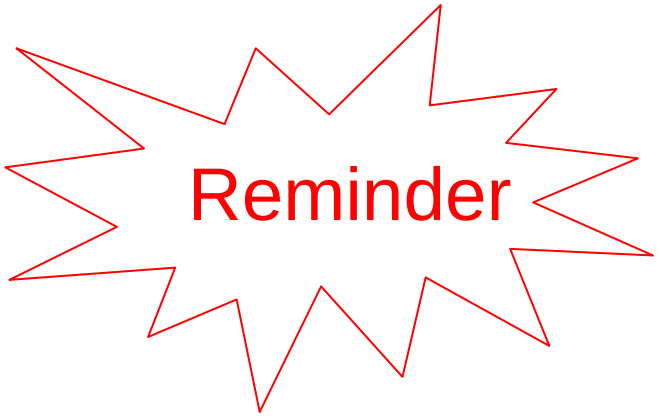
Repository split: reminder, discussion and decision

19/11/2014

P. Hristov

History

- The idea came during the preparation for QM2014
- Jan Fiete showed the first presentation on 10/02/2014 during the weekly offline meeting
- We had more detailed presentation on 17/02/2014 during the Computing board, followed by long discussion and decision to discuss with the physics board the right moment for the repository split
- The Physics board recommended to do the split after QM2014 (end of May)
- We are ten months after the initial discussion, and some concerns came back



Proposal: Factorizing PWGs out of AliRoot

Alina Grigoras
Jan Fiete Grosse-Oetringhaus
Peter Hristov

Computing Board, 17.02.14

Motivation

Initial

Aim: Increase analysis turn around

(= time between code ready and data processed)

1. Increase frequency of code deployment to the Grid

2. Simplify committing of user code to the repository

Current share of analysis activity:

66% LEGO analysis trains vs 32% individual analysis

Final aim: Code ready when leaving the office (evening), full data processed when back in the office (morning)

More on motivation

- Analysis and Core are different in
  - Development patterns
  - Development speed
  - Code stability
  - Code size
  - Release procedure and validation
  - Access rights
- We can profit from this exercise to start the factorization of other AliRoot parts

More on motivation

- The splitting is a step towards the “FairRoot way” of code distribution
  - External software: Root, G3, G4, Boost, CGAL, FASTJET, 0MQ + Pythia6, Pythia8, Hijing, DPMJET, Herwig, TEvtGen, etc generators currently in AliRoot
    - The versions of each package are fixed in a configuration file telling how to download the code: CVS, SVN, Git, tarballs and so on
  - Core software: base classes, steering, detectors, etc.
  - Analysis software: 9 PWGs + few other modules
- The splitting triggered reimplementation of the AliRoot build system using “native cmake”

More on motivation

- Smaller daily distributed analysis tarballs
- Possibility to
  - extend the developers community;
  - establish workflow;
  - improve the code quality;
  - delegate responsibility;

Proposal for First Step

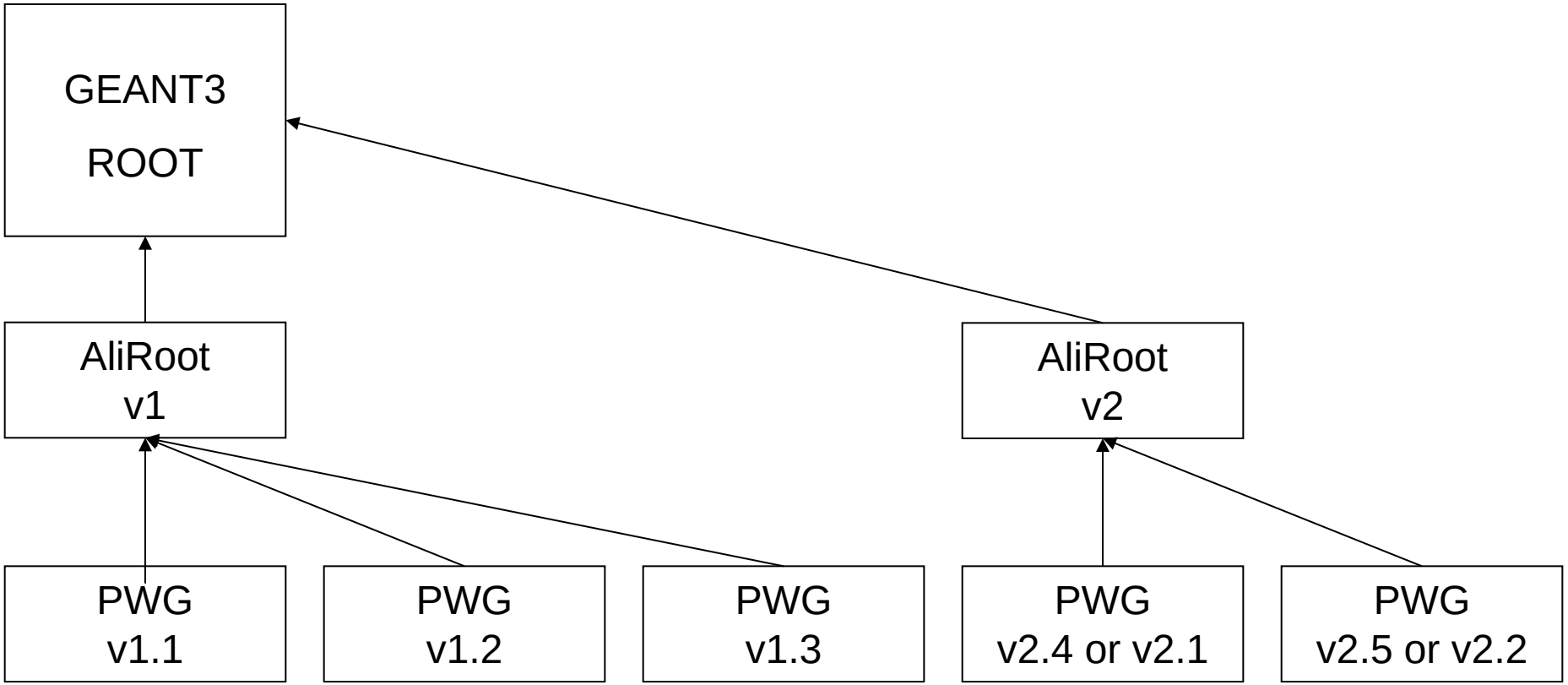
- Factorize PWGs (9 folders) out of AliRoot
  - Separate repository & tagging
- Aim
  - Stable core libraries (ESD, AOD, ANALYSIS etc)
  - Stable supplier objects (PID, physics selection etc)
  - Tag only the parts which changes often (PWGs)
    - Allows further improvements (see below)
- Implications
  - One more dependency in the chain
  - Users need to download and build one more package (at run time it will look identical)
- First step in the modularization of AliRoot

More on splitting

- We can move to AliPhysics also:
  - ANALYSIS or part of it (physics selection, centrality, PID-dependent parts, calibration-dependent parts, TENDERsupplies, ESDfilter, classes for cuts, macros, etc.)
  - OADB
  - Detector calibration macros that use the analysis framework
  - CORRFW
- These modules probably have slower development cycle, but logically belong to AliPhysics

Tagging

- Separate tagging, for example
  - AliRoot for example monthly, or on request
  - PWG twice weekly (like AN tags now): we have it now daily



More on the tagging and dependency management. Questions & objections

- How do I know that AliPhysics YYY requires AliRoot XXX? This will create a lot of mess.
  - The name of the AliPhysics tags can be self descriptive: XXX.YYY and then clearly you see you need AliRoot XXX for AliPhysics XXX.YYY
  - The versions of AliRoot can be specified for each (tagged) version i.e. in the file ARVersions.h and then the CMake function FindAliRoot in AliPhysics can always check if the build is possible (using >= requirement)

Questions and objections (cont.)

- How do I make sure that the changes in AliRoot are taken into account in AliPhysics? If I work with an old AliRoot tag, I may need many fixes to move to the next one.
  - This is the same issue as with Root: when we decide to move to a new version, we usually have to fix some issues
  - We also can use the validation cluster to build AliPhysics master against AliRoot master and automatically detect issues

Questions and objections (cont.)

- AliRoot depends on AliPhysics
  - No, there is no such dependency. The fact that we call macros that load the AliPhysics libraries for AOD/delta AOD production is not dependency. In the same way Root doesn't depend on AliRoot even if we load all our home-made code
- How do we tag AliPhysics? We need stable tags and it is not possible to have them from the master
  - This is not exactly true: we can use a tag from the master and test if the AOD production works. If it doesn't, we follow the same procedure like now: we switch off the task and ask the authors to fix it. Not all the task participate in the AOD creation.

Questions and objections (cont.)

- Who will maintain the compatibility between AliRoot and AliPhysics:
  - For the moment PH-AIP-SDS will take care, but we expect help from the PWGs
  - We also will automatize as much as possible the validation procedures

Future Steps

- Given a factorized PWG package, one can imagine the following improvements:

- Increase frequency of PWG tags
  - Nightly tags & builds

- PWG Repository to which analyzers can commit directly
  - With automatic checking
  - No delay between code ready and in repository

Open Repository

- Current workflow
  - Users send their code by email to expert (PWG responsible or conveners)
  - The expert imports the code, compiles, loads libraries; if successful commits
  - No functional checks are done; Why?
    - Expert has to handle code of tens of users
    - Crashing user code affects only him/herself (train test catches this before submission → user is excluded)
- Can we automatize this?
  - Saves time of users and experts

Open Repository (2)

Incomplete list of ideas on the implementation

- Possibility for the user to provide a code update
  - Build server imports the changes
  - Compiles, loads libraries, (run some code?)
  - If ok → commit ; if error → inform user
- Implementation possibilities are
  - Email target or web page to which patch is sent
  - Direct commit, build server reverts on failure
  - Server-side user branches
- Access control
  - PWG / directory / file level
  - Expert may need to be involved if user wants to add/remove something from a library

More on the open repository

- The build of the AliPhysics will be very fast since the inter-analysis dependencies are small. It is not like a change in AliESDTrack triggers full recompilation of the reconstruction and analysis
  - Possibility to have hooks in CMake, but this needs investigation
- Improved workflows can be adopted in AliPhysics

Some Comments

- Advantages
  - Stable CORE part
  - Regular PWG tags lead to faster build & smaller archive
  - Committers to PWG don't need to update/pull CORE part on each commit/push
- Implications for users
  - Need to download one package more
    - Detector developers may not need PWG package
  - Need to build one package more
    - May build into same directory, no change in includes, library path required
    - AliRoot download script can do this transparently
- Implications for train operators
  - Selection of PWG tag instead of AliRoot tag, dependencies automatic

17/02/2014

Discussion

- Proposals discussed in last week's Offline mtg and on mailing list
- Generally agreement on the concept
- Various ideas for the implementation (some included already today)
- Different opinions on what to put in this library and on the naming
- Proposals for more fine-grained splitting of AliRoot (→ foreseen for AliRoot 6 / RUN 3)
  - Ideally clear separation of responsibilities: external packages; framework for simulation, reconstruction, analysis; PWG-specific analysis code
  - Account for the different development cycles
  - Different distribution patterns (i.e. DAQ doesn't need the simulation and analysis parts of AliRoot)

  → PWG part is only first step in the modularisation of AliRoot
- More complicated set of dependencies
  - Possible problems in case of development based on old "AliRoot" versions

Current status

- Reimplementation of the CMake build system (Alina)
  - The libraries and binaries are almost complete
  - The rootmap files are generated
  - Fixes for circular dependencies are needed
  - The creation of DA: ongoing
  - Creation of PAR files: not yet started
  - Code checker: obsolete, has to be replaced or revived
  - Documentation: ongoing. We would like to move to Doxygen, can be done after the split
- Repository split: tested (Alina, Dario)

Current status

- Adapt build server (Alina): will be done when the repository is locked
- Adapt train system (JF): will be done when the repository is locked
- Adapt MonALISA scripts (Alina, Costin): will be done when the repository is locked
- Adapt AliRoot download scripts (Dario): done
- Adapt documentation (Dario): almost done

Proposed decision

- Follow the earlier decision and provide the two repositories next week
- Implement gradually the new ideas