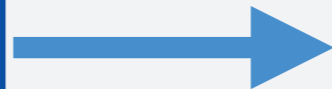# CMake for Geant4 10

Ben Morgan

Gunter Folger

Since Chartres...

- **Install/reuse Data Libraries**

- **Allow C++ Standard selection**

- **Multithreaded build**

- **Bug fixes/tweaks**

g4build — cc

ccmake

```
CMAKE_BUILD_TYPE                          Releas
CMAKE_INSTALL_PREFIX                      /usr/l
CMAKE_OSX_ARCHITECTURES
CMAKE_OSX_DEPLOYMENT_TARGET
CMAKE_OSX_SYSROOT
GEANT4_BUILD_MULTITHREADED                OFF
GEANT4_INSTALL_DATA                       OFF
GEANT4_INSTALL_DATADIR
GEANT4_USE_G3TOG4                         OFF
GEANT4_USE_GDML                           OFF
GEANT4_USE_INVENTOR                       OFF
GEANT4_USE_OPENGL_X11                     OFF
GEANT4_USE_QT                             OFF
GEANT4_USE_RAYTRACER_X11                  OFF
GEANT4_USE_SYSTEM_CLHEP                   OFF
GEANT4_USE_SYSTEM_EXPAT                   ON
GEANT4_USE_SYSTEM_ZLIB                    OFF
GEANT4_USE_XM                             OFF

CMAKE_BUILD_TYPE: Choose the type of bui
Press [enter] to edit option
Press [c] to configure
Press [h] for help              Press [q] t
Press [t] to toggle advanced mode (Curre
```

CTest - Integration Testing

- **Integration testing set-up stable**

  - Updates to tested platforms, including MT builds

  - Streamlined scripts

- **Open Problems**

  - Still occasional problems with updates from SVN

  - Reliability of CDash server

CDash - Continuous and Nightly Testing

- **Shift system working well**

  - geant4tags to select/accept/reject tags

  - [CDash dashboard](#) for viewing test results

  - Edupad page for logging by shifter

- **Continuous builds help filter out "bad" tags quickly**

  - Reduced set of tags used to continuously test proposed and selected tags

  - *You should check CDash Continuous after proposing a tag!*

Updates for 10.0

- **Default usage of all builtins:**

  - CLHEP, expat, zlib etc

- **Never build examples**

  - Just install source code

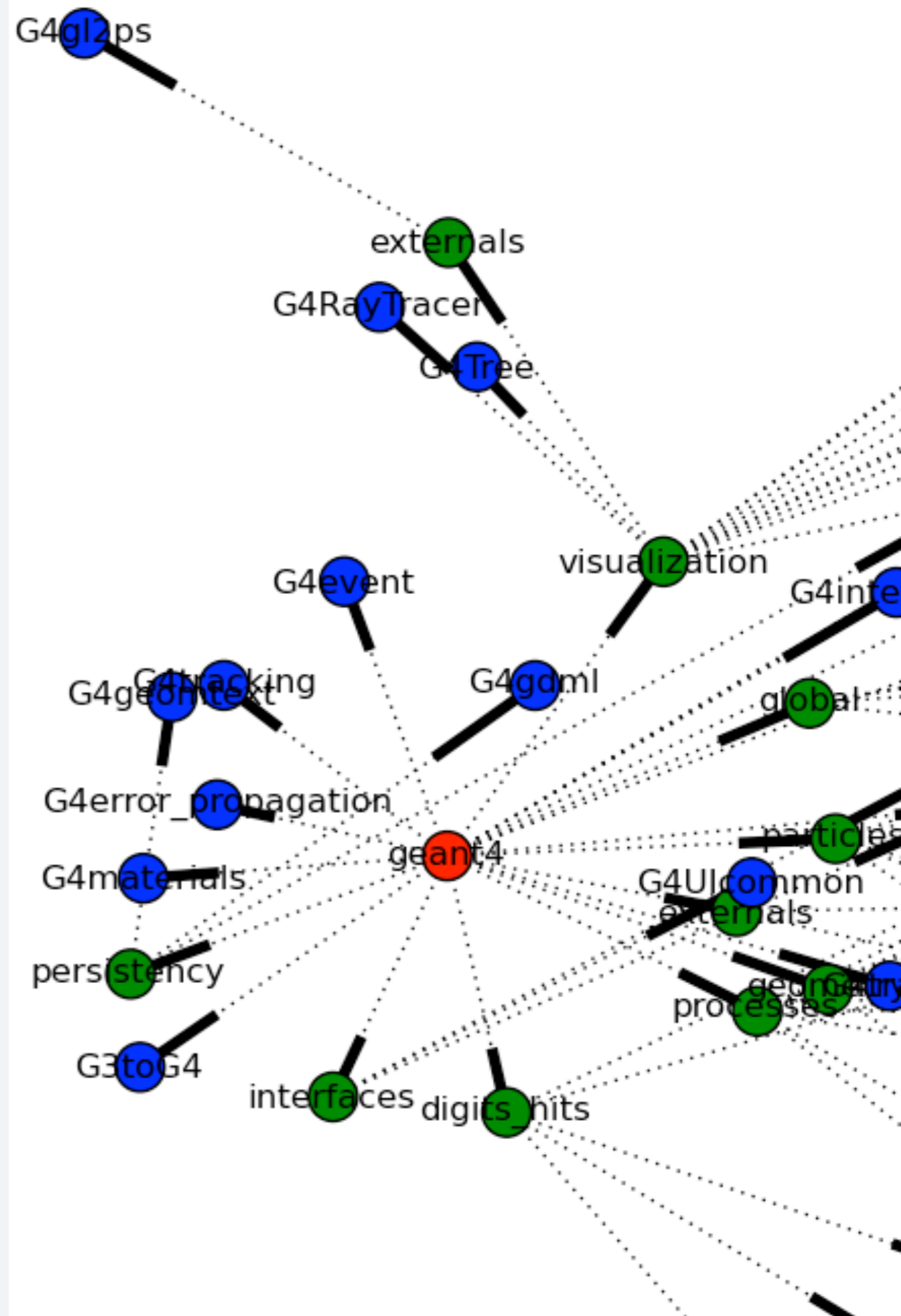- **Documentation**

GNUMake Support

- **Not retired in 10.0, but simplify system**

- **Demo available for discussion this week**

  - Environment variables become Make variables.

  - One line change to User GNUmakefiles

Geant4/CMake Proposed 2014 Development

Developer support

Library modularization

jira-geant4.kek.jp/browse/DEV-81 — [DEV-81] Improve CMake API for, and modularization of, Geant4 Categories – Geant4 Requirements Tracker

[DEV-81] Improve CMake API for, and modularization of, Geant4 Categories – Geant4 Requirements Tracker   Installation and Configuration

Development / DEV-81

Improve CMake API for, and modularization of, Geant4 Categories

✏ Edit   💬 Comment   Assign   More ▾   Start Progress   Resolve Issue   Workflow ▾

Details

Type:	☑ Task	Status:	→ Open (View Workflow)
Priority:	↑ Major	Resolution:	Unresolved
Affects Version/s:	10.0	Fix Version/s:	10.0
Component/s:	Code Reviews and Improvements, ... (2)	Security Level:	Developers only (Developers only)
Labels:	None		
Environment:	All		

People

Assignee:	👤 Ben Mor
Reporter:	👤 Ben Mor
Votes:	0️⃣ Vote for
Watchers:	4️⃣ Stop wat

Dates

Due:	20/Sep/13
Created:	10/Jul/13 3:3
Updated:	22/Jul/13 12:

Agile

View on Board

Description

The current "sources.cmake" file mechanism for developers to declare the sources/resources and dependencies of their code in Geant4's CMake system requires duplicate information to be provided, and could be more coherent. The main issues are outlined here:

https://indico.cern.ch/contributionDisplay.py?sessionId=2&contribId=79&confId=199138

Improvements here need to cover these issues, and also address the need for a better modularization of the Geant4 libraries. This latter issue comes from the user requirement to be able to switch off certain physics processes/models at build time. A further benefit is to maximize coherence of each component library whilst minimizing dependencies.
Neither the known granular (too many decoherent libraries) nor global (too few large libraries) builds provide a singular solution, but a dual mode build will not be supported due to time limits.

Implementing a new scheme first requires an analysis of the dependency graph of the Geant4 granular libraries. The aim here is to identify:

1) Libraries that may benefit from merging
2) Libraries that may benefit from splitting
3) Direct dependencies of each library to reduce overlinking
4) Public and private headers of each library

What do YOU want from the CMake system?

Geant4 Development Pocket Guide Draft

Ben Morgan   Friday 20th September, 2013

Contents

https ⬤ geant4.cern.ch/collaboration/svnmanage

Geant 4

Code Management

This is a quick-reference guide for Geant4 developers who need to access to the Gea

Please also look at the Tag & Release Policy document for Geant4.
For a more exhaustive tutorial on SVN, see documentation you find from the original

The Geant4 SVN repositories are located on the svn server svn.cern.ch. There are
g4tests), and Geant4-related tools (in g4tools). The repository is accessible only th
collaborators after signing-in once a year to confirm their Collaboration membership a
access the form, contact your working group coordinator to get you onto the list of col

1. The GEANT4 SVN repository structure
2. Browsing the Geant4 svn repository
3. How to setup and access the repository
4. Checkout and update the code
       1. Checkout of User Documentation
5. Viewing changes and resolving conflicts
6. Committing changes to the repository
7. Tagging and versioning
8. Useful SVN commands
9. Information on svn
10. Where to find released source code

## 1. The Geant4 code and SVN repository structure

The Geant4 source is actually structured in the source directory under *geant4* in a tre
tests, etc. The source is distributed according to each sub-domain.
The sub-domain *global* is a place holder for all the development done which affects al
include directories contain header files (.hh) and inline functions definition files (.icc);
related sub-domain, to test the code under distinct conditions.

The Geant4 code repository has three top level directories, trunk, tags, and branches

Do you want more documentation?
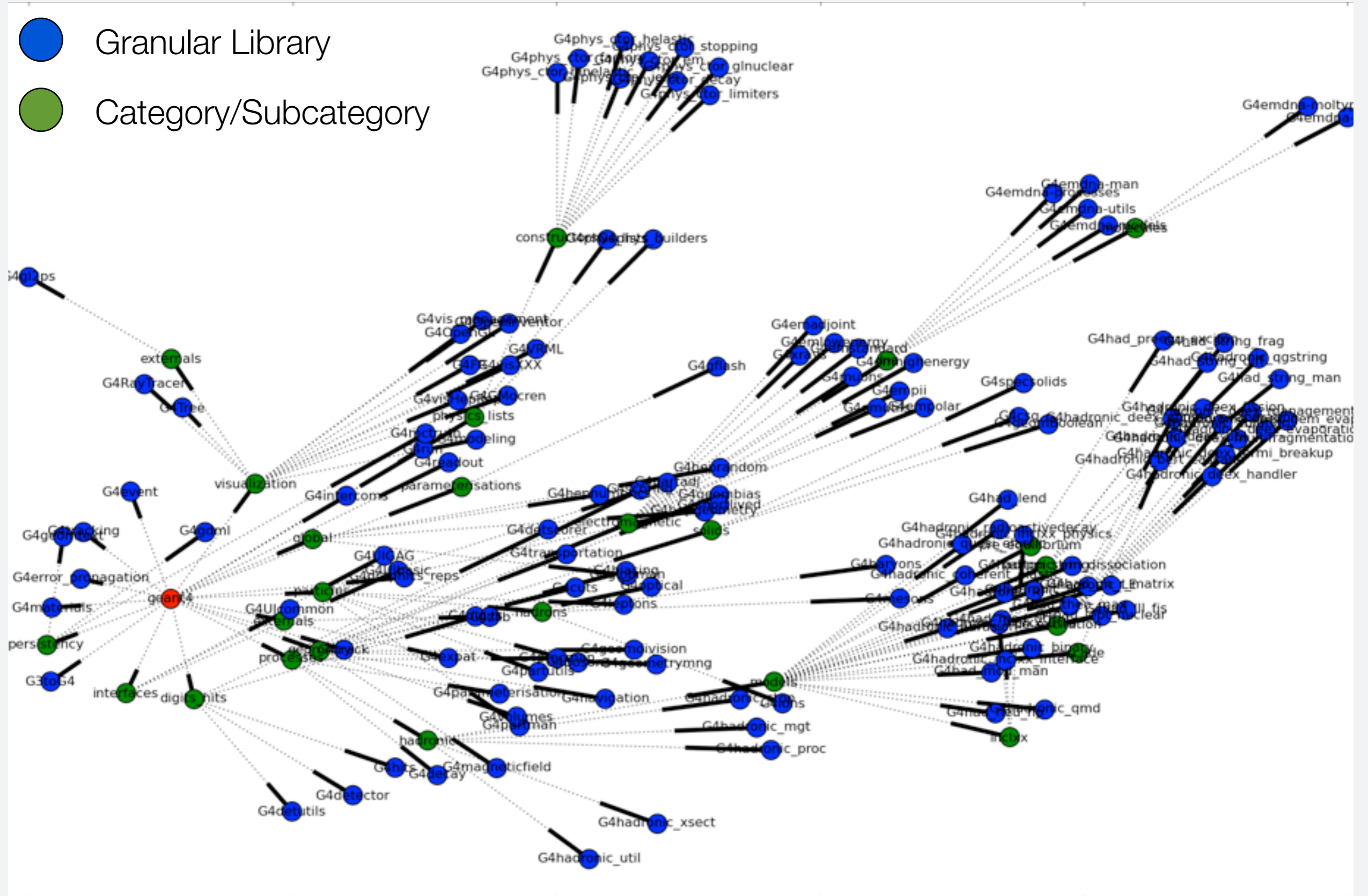
How best to organise and build these libraries?
Ease of maintenance, development and performance

Granular
Too Many
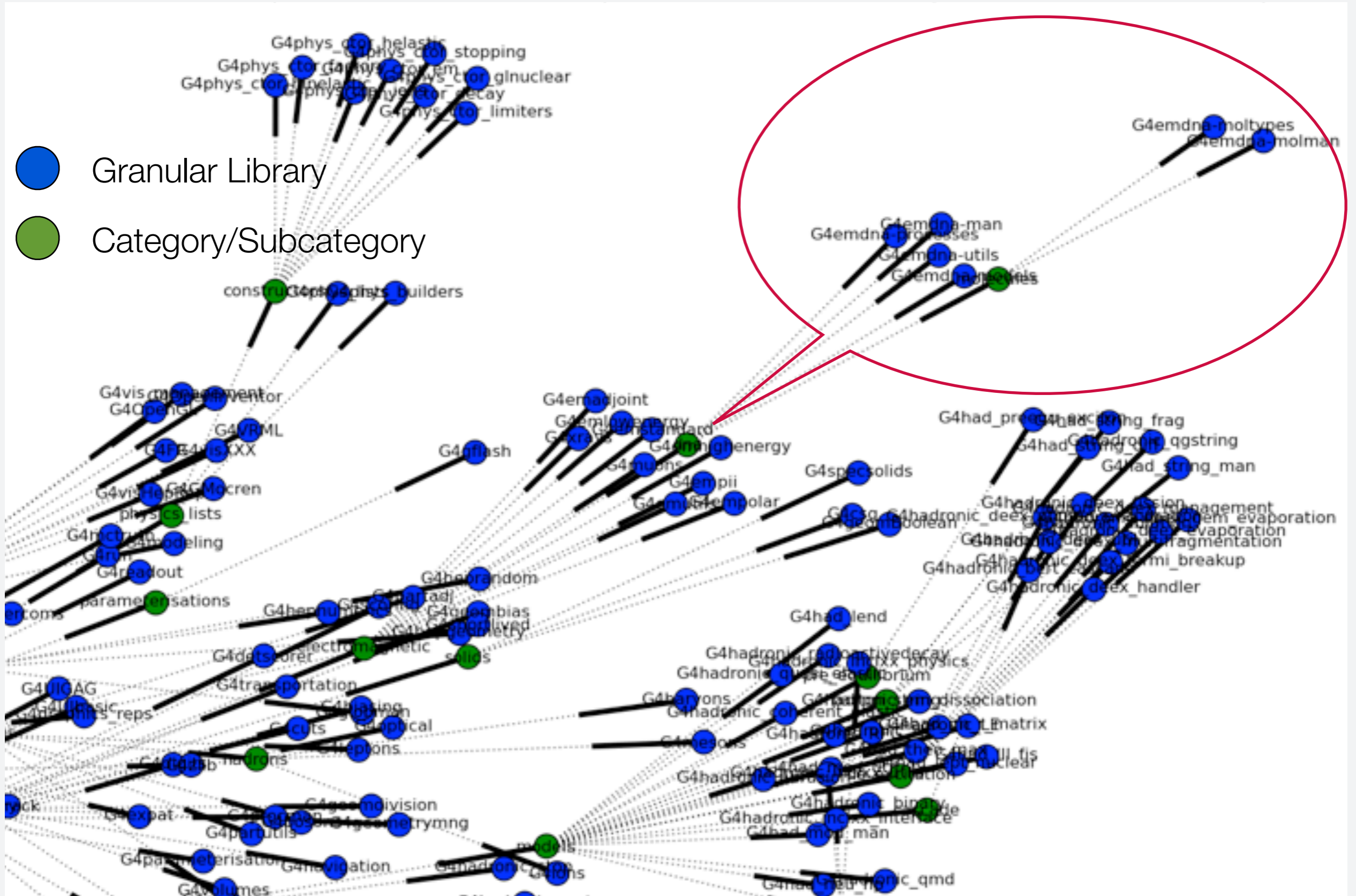
Global
Too Big

Merge
↓
Optimum
⇈
Split

Our current solutions are suboptimal, and make life difficult for you

Example: DNA
Not always required, BUT, are 6 libraries really needed when it is?

3205
headers

986
private to granular
library
(estimate)

1983
detail of granular
library
(estimate)

Hide headers of implementation details - Cleaner Public API

A Quick Summary

- **CMake and Testing on course for 10.0 Release**

  - Please let Software Management of any further requirements ASAP!

- **Discuss CMake updates for 10.0**

  - CMake defaults for user installs.

  - Legacy GNUmake support

- **Discuss proposed modularization work for 2014**

  - **This covers the whole toolkit, so your input is crucial**