Updates on CMake, Modularization, and C++17

Ben Morgan, Gabriele Cosmo, Gunter Folger, Jonathan Madsen

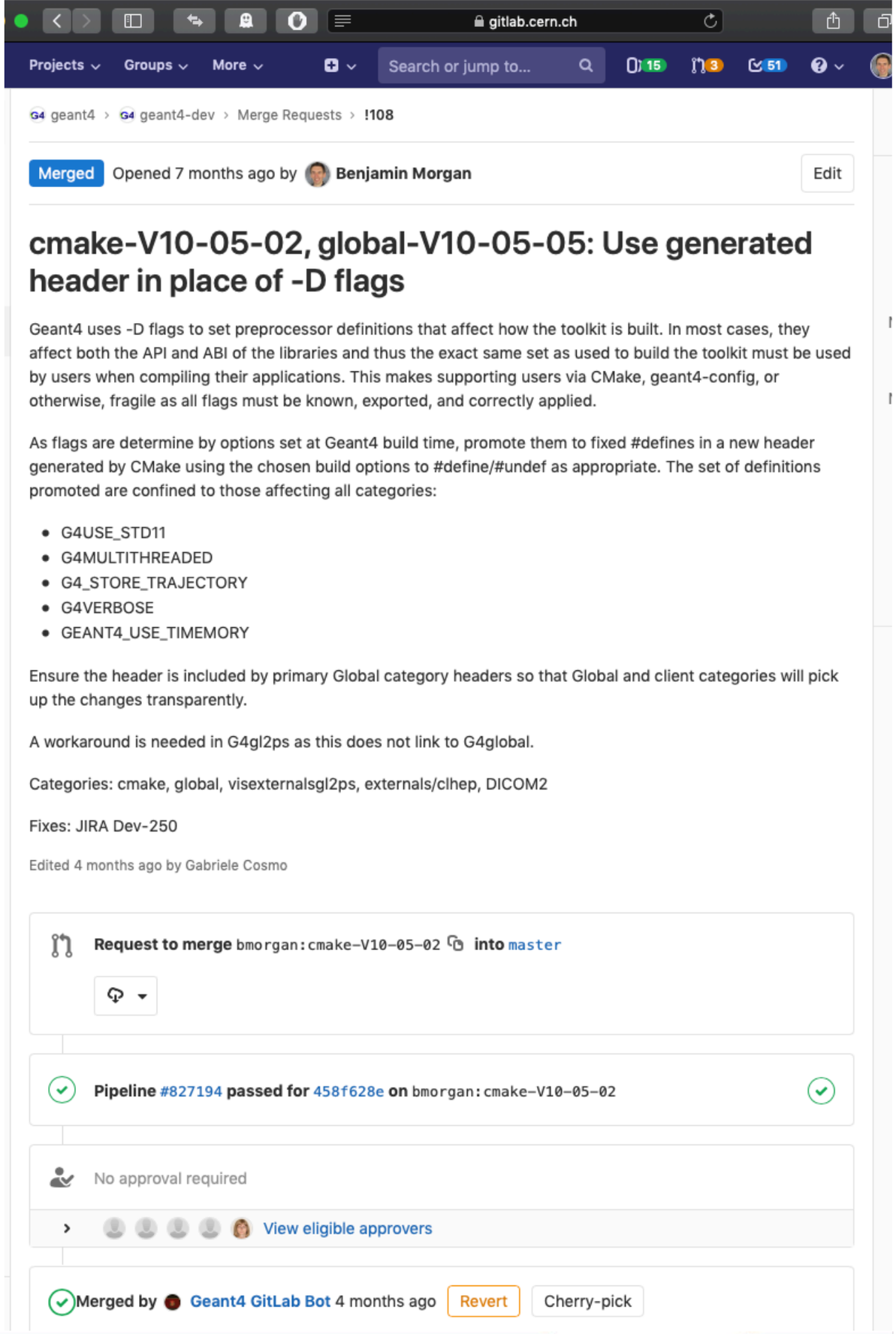
WARWICK
THE UNIVERSITY OF WARWICK

CMake: Minor Updates

- Mostly bug fixes:
  - *Addition of missing* `GEANT4_USE_SMARTSTACK` *option*
  - *Removed -DG4FPE_DEBUG from main DEBUG build mode due to incompatibilities with Qt*
- *New "DEBUG_FPE" mode for debug with FPE testing*
- *Support use of legacy OpenGL over GLVND*
- *Fixes in Geant4Config for compiler flag, path, and configuration errors reported via Bugzilla*

MR !108: Replacement of Core -D flags

- Ensures required global level definitions are consistent when building and using Geant4
- Eases configuration and portability
- No impact on Geant4 code, User code must include G4Types.hh to pick up definitions

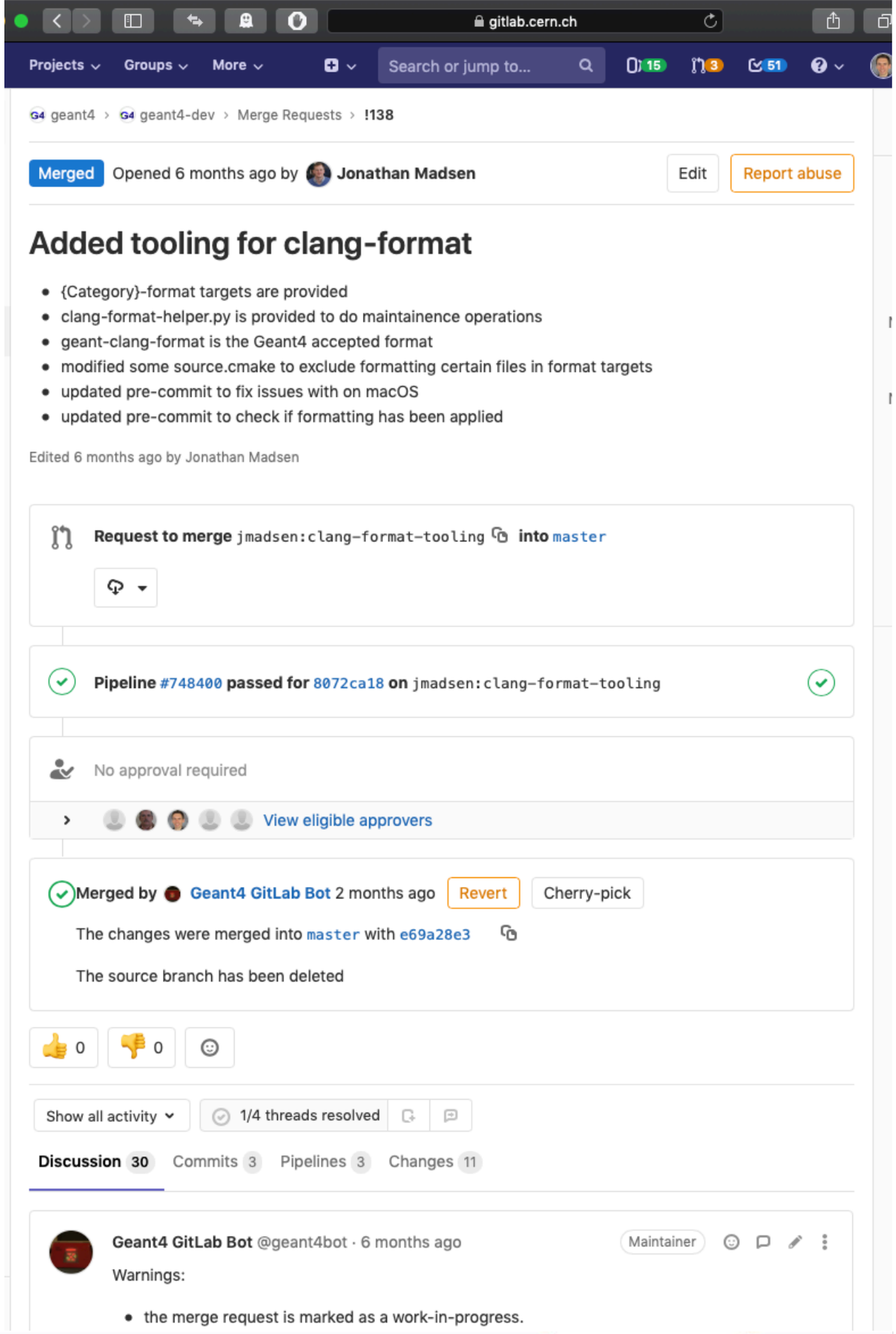


MR !264: Replacement of Geometry -D Flags

- Same purpose as for Global -D flags
- No changes required in Geant4 or User code
- Some residual -D flags are still propagated from VecGeom
  - *Ticket raised on VecGeom JIRA to address this: https://sft.its.cern.ch/jira/projects/VECGEOM/issues/VECGEOM-535*

MR !138: Support use of clang-format

- Allows automatic code formatting when clang-format is available via, e.g.
  - `make G4global-format`
- Totally optional, but could be automated
- Style matches current Geant4 formatting



MR !138:Excluding Code

- May be parts of your code you don’t want formatted
  - *e.g numeric tables, certain chains of statements*
- You can disable formatting of any block by bracketing it between “`// clang-format on|off`” comments:

```cpp
#include <optional>

// This will be formatted
std::optional<int> GetAnswer(int r) {
  if ( some_condition_on(r) )
    return 42;

  return {};
}

// This will be left untouched
// clang-format off
auto result = GetAnswer(someInteger);

if (result)
  PrintResult(*result);
else
  Report("no result obtained");
// clang-format on
```

Avoiding Parkinson's Law of Triviality

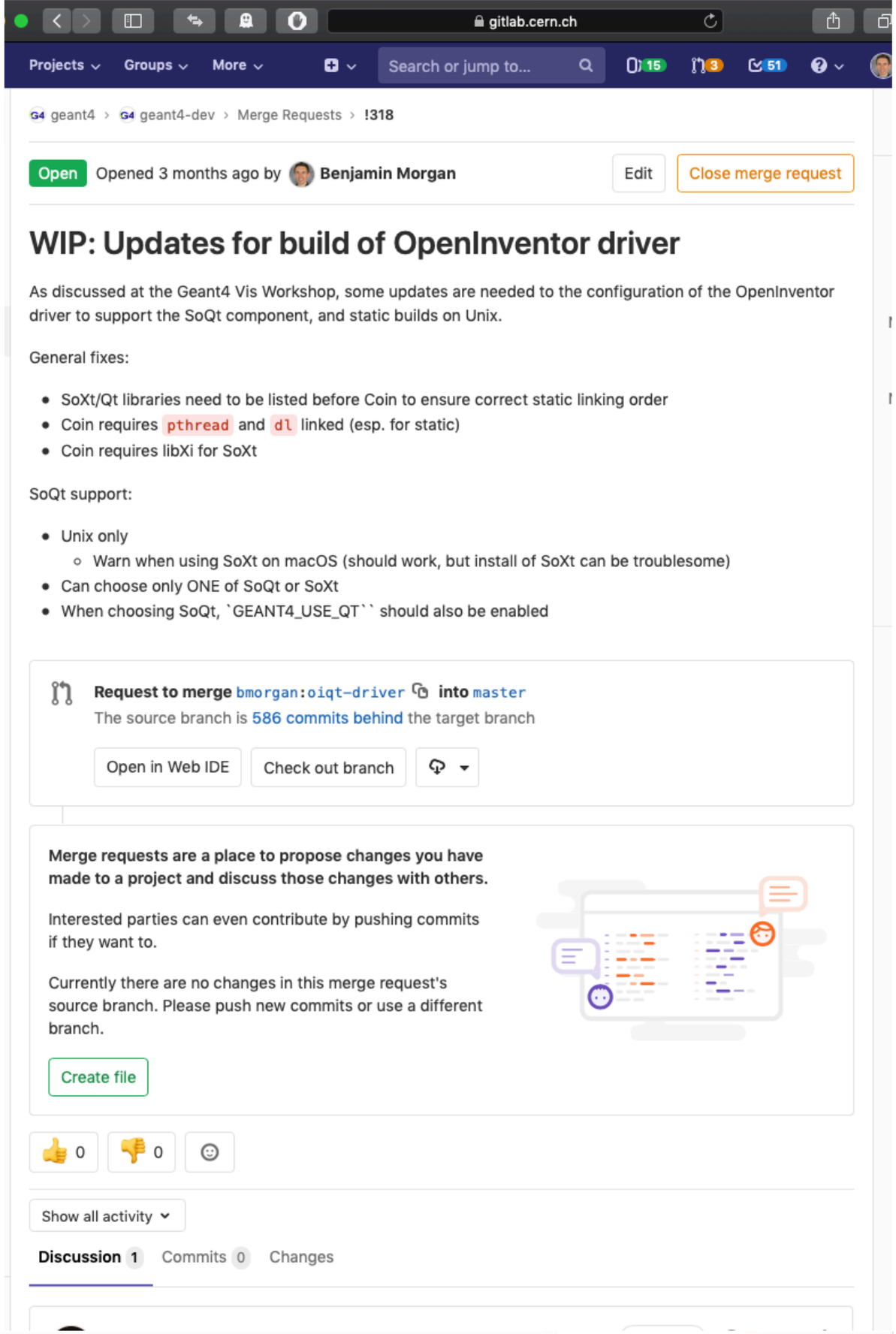
Style is a classic bikeshed.
Use for what it's good for:
consistency, maintainability

WIP: CMake Minimum Version Update

- 10.6 will require 3.12 for building, at least 3.8 if building applications with CMake
  - *Fully support Windows DLL builds without workarounds*
  - *Fully support C++17 builds*
  - *Improved support for finding, using, and exporting third party packages (e.g. zlib, Qt)*
  - *Better support for newer platforms*

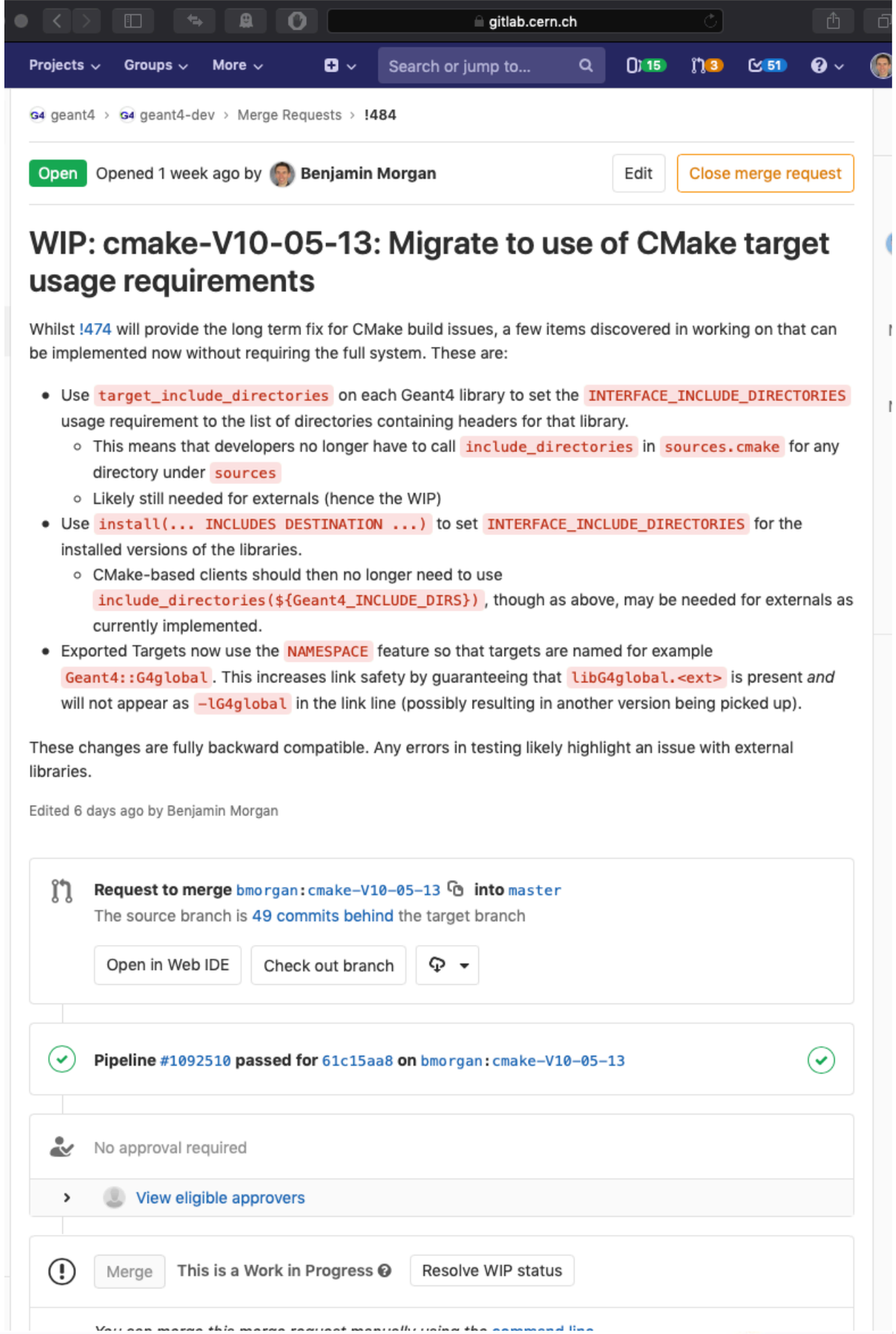
WIP: MR !318 on OpenInventor

- Several fixes identified during Vis mini workshop
- Main complexity is tracking interdependent options and build/use time.
- See later item on use of third party packages in general.



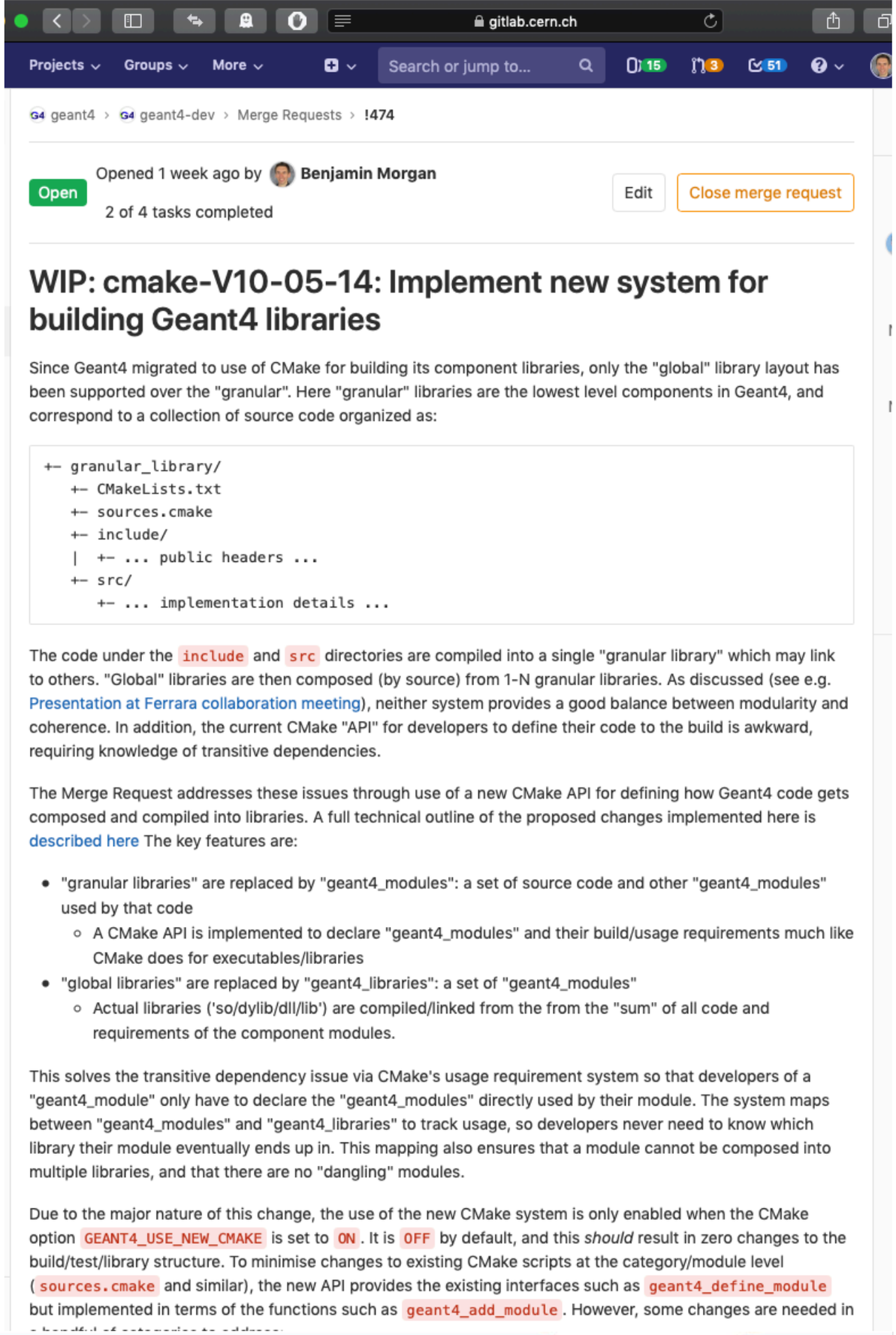
WIP: MR !484 on Usage Requirements

- Backporting three items of functionality from new CMake system (MR !474)
- Remove requirement for `include_directories` in your sources.cmake
- Full deployment of **usage requirements**
- Improved link safety with target “namespaces”



WIP: MR !474 on New Library Build Model

- Implementation of **system presented at Ferrara Meeting**
  - *Delayed by Git work*
- Developers declare:
  - *Code in this (sub)category (include/src)*
  - *(Sub)categories used by this one*
- "Integrator" declares
  - *Composition of (sub)categories into libraries*
- Single way to create libraries, but can be modified easily without impacting developers



Changes you will need to make after MR !474

- To begin with, **NONE**
  - *System is backward compatible with existing code in your* `sources.cmake` *scripts*
- Initial library structure identical to “Global” to allow comparative testing
- `sources.cmake` migration over several weeks and Merge Requests (Git helps immensely here!)
  - *System will also be fully documented through Sphinx*
- **Input and Feedback on MR !474 are welcome!**

Build Topics after 10.6

- These are to motivate discussion amongst you this week and for next year's workplan

- *Use, support, and deprecation of Third Party Libraries in Geant4*

- *Choice of minimum C++ Standard and Platforms to support in 2020 release*

Third Party Library use in Geant4

- Primarily used in Analysis (HDF5, FontConfig) and UI/Vis (Qt, OpenGL, Inventor …), Geometry (Xerces/VecGeom) categories
  - *This is a **good** thing - we shouldn't reinvent the wheel, but…*
- Try and avoid exposing Third Party headers in Geant4 headers. This makes building/deploying applications harder.
- **We should not be afraid of removing components of Geant4 that depend on EOL, obsolete, or otherwise unsupported packages**

Migrate to C++17 in 2020?

- **Question**: Should we make the 2020 release require the C++17 Standard as the minimum?
  - *See **Presentation at Lund** on language features added by C++17*
- ***Benefit*** would be use of these new features to help improve and simplify Geant4 for users
- ***Cost*** would be dropping support for older platforms/toolchains
  - ***https://en.cppreference.com/w/cpp/compiler_support***
  - ***Personal** opinion: I think we go for it through a Major Release, core/ common set of platforms are fine directly or with easily available tooling*

Discussion

- Work on modularisation system underway
  - *Comment/Review/Feedback under MR !474 are very welcome*
- **Two build related questions for you to discuss now and over the coming days:**
  - *Better management and deprecation timeline(s) for third party libraries used in Geant4*
  - *Should we move to requiring C++17 for the 2020 release?*