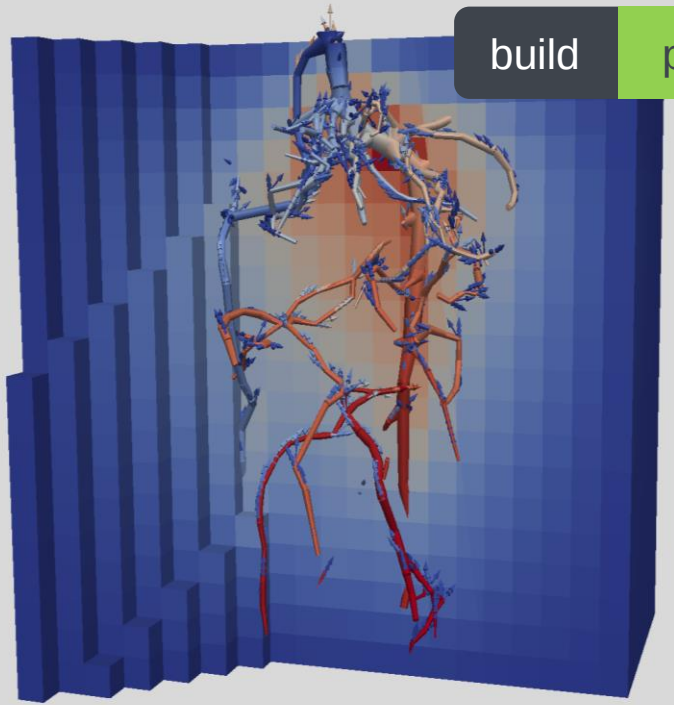
University of Stuttgart
Germany

Computational Methods in Water Resources 2016
20th – 24th June 2016
University of Toronto, Canada



build passed

Automated system testing in scientific numerical software frameworks

using the example of Dune / dune-pdelab / DuMu^{X}

Timo Koch[1]
Dominic Kempf[2]
Bernd Flemisch[1]
Peter Bastian[2]

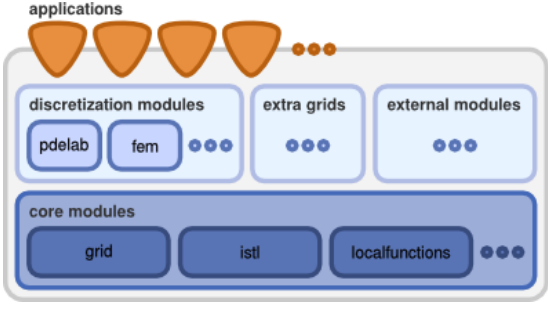
[1] University of Stuttgart, [2] Heidelberg University, Germany

Background

DuMu^X and DUNE

- DUNE – a numerical software framework for solving PDEs
  - Developed at over 10 universities in Europe
  - Open-source development model
  - Highly *modular*; loosely connected modules
  - Template-based C++ programming

- DuMu^X – application module, porous media simulator
  - Modular structure
  - Material framework / laws; fluid systems
  - Non-isothermal multi-phase multi-component models
  - Cell- and vertex-centered finite volume discretization



get Dune / Dumux at

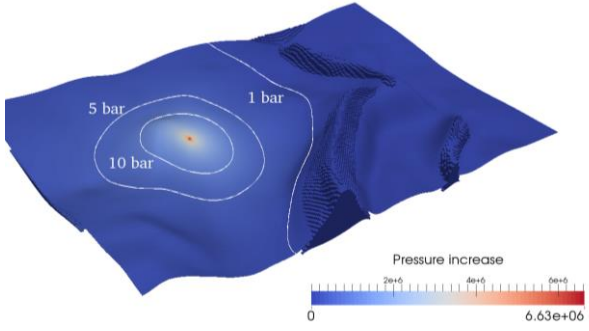
https://www.dune-project.org/
http://dumux.org/

repositories at

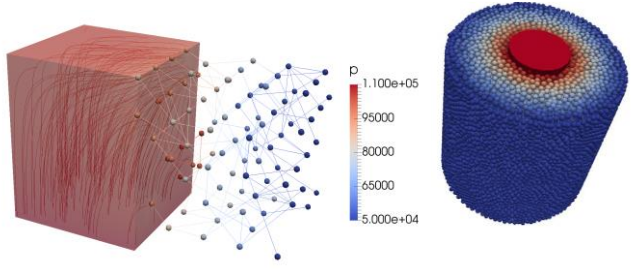
https://gitlab.dune-project.org/groups/core
https://git.iws.uni-stuttgart.de/dumux-repositories/

DuMu^x – DUNE for Multi-{Phase, Component, Scale, Physics, ...} flow and transport in porous media

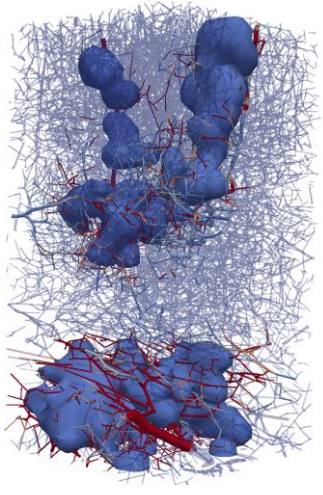
Groundwater contamination (Alexander Kissinger)



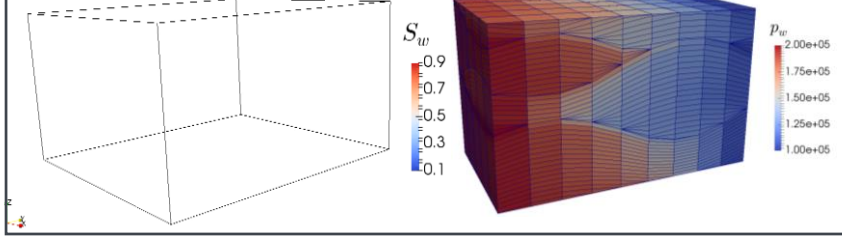
Porenetwork – Darcy coupling (Kilian Weishaupt, T. K.)



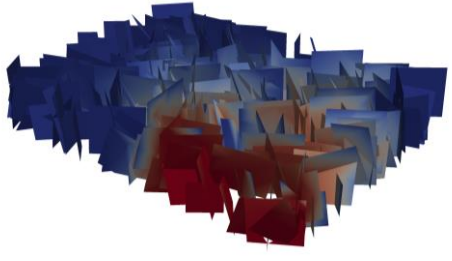
Capillary networks 1D-3D coupling (T. K.)



Two-phase flow on cornerpoint grids with NL-TPFA (Martin Schneider)



Discrete fracture networks (Dennis Gläser)



University of Stuttgart

Supervisors: Bernd Flemisch, Holger Class, Rainer Helmig

Motivation

Why testing and how?

Motivation

Why is testing necessary and important?

- Open-source / research code is under continuous development (bugs appear!)

- We want
  - *Reproducible* and *trustworthy* numerical results as basis for publications
  - *Sustainable* code development (*code reusage, combining codebases*)
  - Increasing trust and transparency, quality assurance

- Main problem
  - Developers are coding and researching (PhD students, professors)
  - Little time for documentation and testing

How to test – overview of different test types

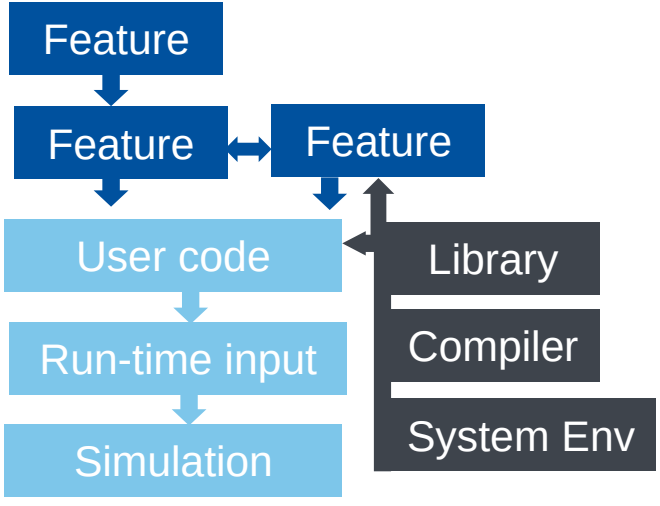
- *Unit testing* (single feature)
- *Integration testing* (few features to functional unit)
- *System testing* (end-user setup, feature combination) → **mostly neglected!**
  - Challenges – high coverage, test evaluation

- Build & run
- Comparison of simulation output with reference output (from stable versions)
- Benchmarks, real-world examples / experimental data (validation)
- Convergence tests against analytical solutions (verification)
- Scalability tests

Motivation

Why is **system testing** necessary and important?

- Frameworks provide combinable features
- They depend on third-party libraries / system setup



- Only unit testing is not enough!
- Benchmarks typically only test a single end-user setup!

- Problem
  - Huge number of possible user setups → combinatoric explosion
  - Hard for generic algorithms to eliminate non-sense combinations

Facilitating system testing

dune-testtools

D. Kempf, T. Koch. *System testing in scientific numerical software frameworks using the example of DUNE*, 2016 (in revision)

repos: https://gitlab.dune-project.org/groups/quality

dune-testtools

Simplifying system testing for code-developing scientists

- Tools simplifying the writing of system tests

- Idea – use common configure files (ini files) with extended syntax (meta ini files) defining a group of tests
- *“One* source file, *one* meta ini file, *one* line CMake”
- Written in Python, customizable, easy scripting

- Test evaluation tools

dune-testtools

Simplifying system testing for code-developing scientists

Regular Dune ini file

```
[TimeManager]
TimeStepSize = 1.0e-3

[Assembler]
PartialReassembly = true

[Grid]
Refinement = 3
```

dune-testtools

Simplifying system testing for code-developing scientists

Dynamic (run-time) variations

```
[TimeManager]
TimeStepSize = 1.0e-3, 1.0 | expand

[Assembler]
PartialReassembly = true, false | expand

[Grid]
Refinement = 0, 3, 5 | expand

({TimeManager.TimeStepSize} == 1.0e-3 and
    {Grid.Refinement} == 5) | exclude
```

Static (compile-time) variations

```
YASP = Dune::YaspGrid<{__static.DIM}>
UG = Dune::UGGrid<{__static.DIM}>

[__static]
DIM = 2, 3 | expand
GRIDTYPE = {YASP}, {UG} | expand
```

dune-testtools

Simplifying system testing for code-developing scientists

CMake build system integration

```
dune_add_system_test(BASENAME uniquename
                     SOURCE mytest.cc
                     INIFILE conf.mini
                     SCRIPT vtucompare)
```

Various test evaluation tools

- The SCRIPT parameter:
  - Python wrapper for custom test *execution* and *evaluation*
  - Customizable – some are implemented:
    - Comparing output ini files
    - Comparing VTK files
    - Convergence test wrapper
    - Parallel testing
    - Just checking exit code

Integrating (system-) testing in the development workflow

Automated builds and open-source workflow

Automated builds and open-source workflow

Integrating (system-) testing in the development workflow

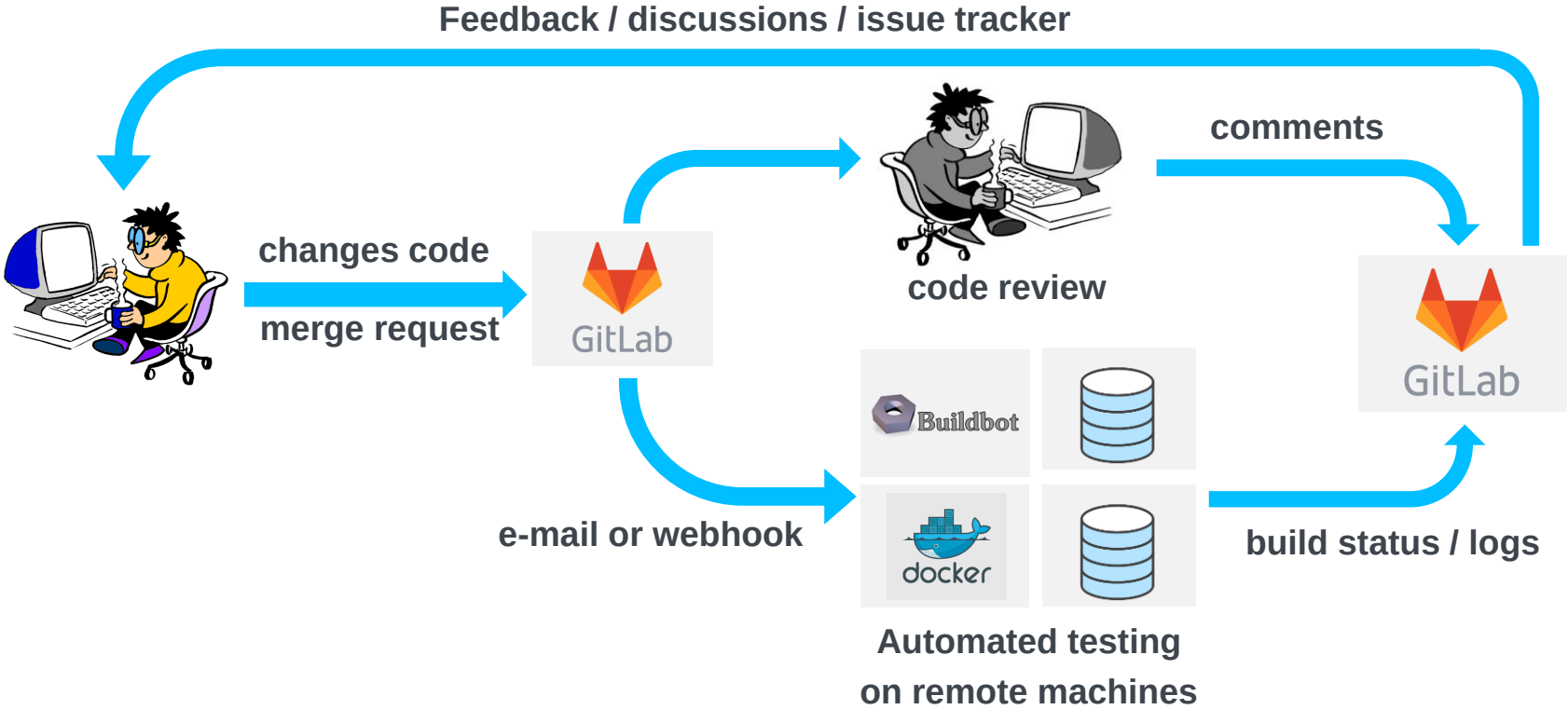
- Git repositories hosted on a GitLab server
- Merge-request based workflow
- Transparent development, issue tracker, contributions
- Integrated Continuous Integration (CI)

- Python framework for Continuous Integration
- Highly customizable (!)
- Communicates with GitLab

- Modelling user lands / system environments
- Robust and highly portable build setups
- Cross-platform

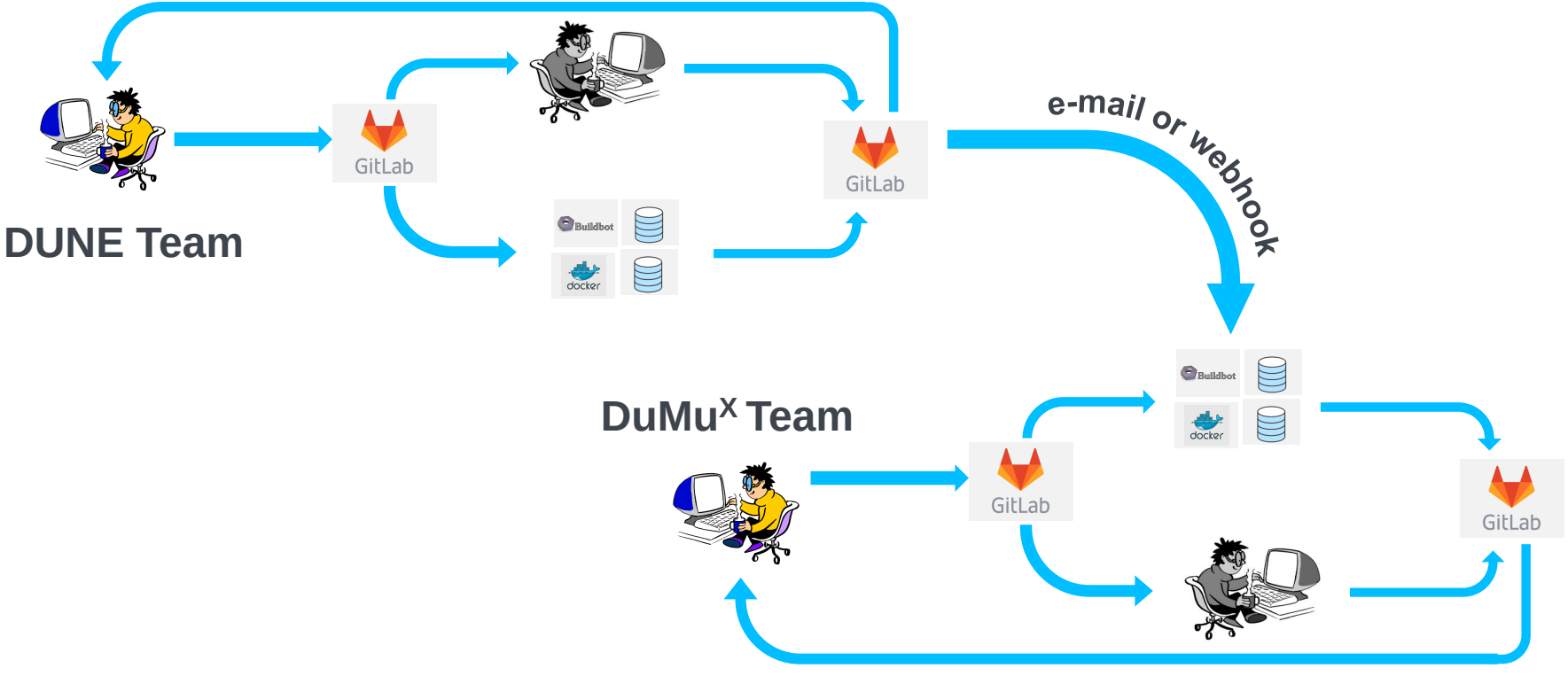
Automated builds and open-source workflow

Suggested development workflow (Continuous Integration)



Automated builds and open-source workflow

Suggested development workflow (Continuous Integration)

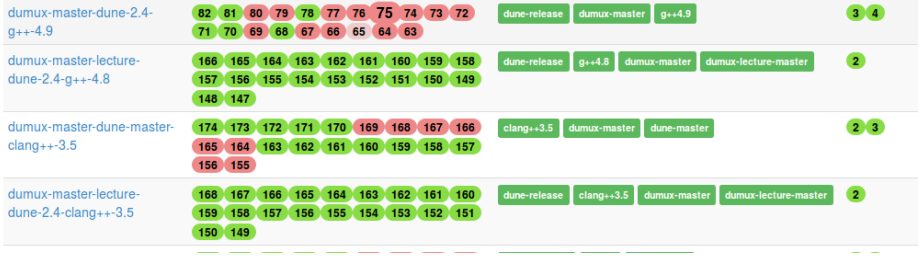


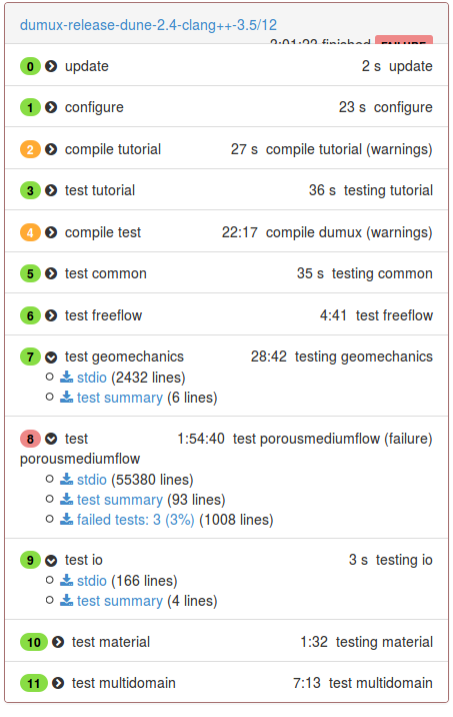
Automated builds and open-source workflow

What did we learn so far?

- Automated testing is detecting early if a bug was introduced
- Bugs can be easily tracked to individual commits
- Fixing / writing tests often reveals otherwise unnoticed bugs
- Leads to improvement of the code base quality
- Makes maintanance easier

visit https://git.iws.uni-stuttgart.de/buildbot/





University of Stuttgart
Institute for Modelling Hydraulic and Environmental Systems

Thank you!

Timo Koch

e-mail timo.koch@iws.uni-stuttgart.de
phone +49 (0) 711 685-64676
fax +49 (0) 711 685-60430

University of Stuttgart
Institute for Modelling Hydraulic and Environmental Systems
Pfaffenwaldring 61, 70569 Stuttgart

The project is supported by

Baden-Württemberg
MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST

Wassernetzwerk Baden-Württemberg

SimTech
Cluster of Excellence

Universität Stuttgart

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386