Observatoire de la CÔTE d'AZUR

Membre de UNIVERSITÉ CÔTE D'AZUR

FargOCA refactoring
Why, How & When

E. Lega & A. Miniussi

Observatoire de la CÔTE d'AZUR
Membre de UNIVERSITÉ CÔTE D'AZUR

Motivation

- ✓ Avoid replication of development effort
- ✓ Add improvement (including perf) more easily, more safely, for more use cases
- ✓ Improve usability and acquisition curve *for new comers*
- ✓ Cut down costs (time is a currency, and machine time is not the most expensive)
- ✓ Fame and Glory (money had to be canceled as a primary objective)

Observatoire de la CÔTE d'AZUR

Membre de UNIVERSITÉ CÔTE D'AZUR

Guidelines

- ✓ Try to merge all feature in one single flexible code
- ✓ Move from a monolithic procedural code to a library based code
- ✓ Do not be emotionally attached to source code (no, that dead code will not be used at some point).
- ✓ Add testing
- ✓ Provide extension points through hooks
- ✓ Prepare a release, even though you do not have customer.

Observatoire de la CÔTE d'AZUR

Membre de UNIVERSITÉ CÔTE D'AZUR

Problem

- ✓ People writing code in a HPC context often lack the basics understanding of software engineering,
- ✓ … yet fail to consider that a problem.
- ✓ After a few year of fork and « temporary » modification, any serious work requires a major cleanup.
- ✓ Restarting from scratch is not an option.

Release

"The minute you've got a product, you're a sitting duck"

- ✓ Even if not “distributed”, the first release is important as it provide a consistent:
  - documentation
  - testing
  - build
- ✓ Limit the temptation to « going back »
- ✓ It won't be complete, lot of cools stuff will be missing -> but it’s not supposed to never be finished.
- ✓ Our first (non feature complete) release will be out "soon" (I know...)

Regarding that « Law » thing...

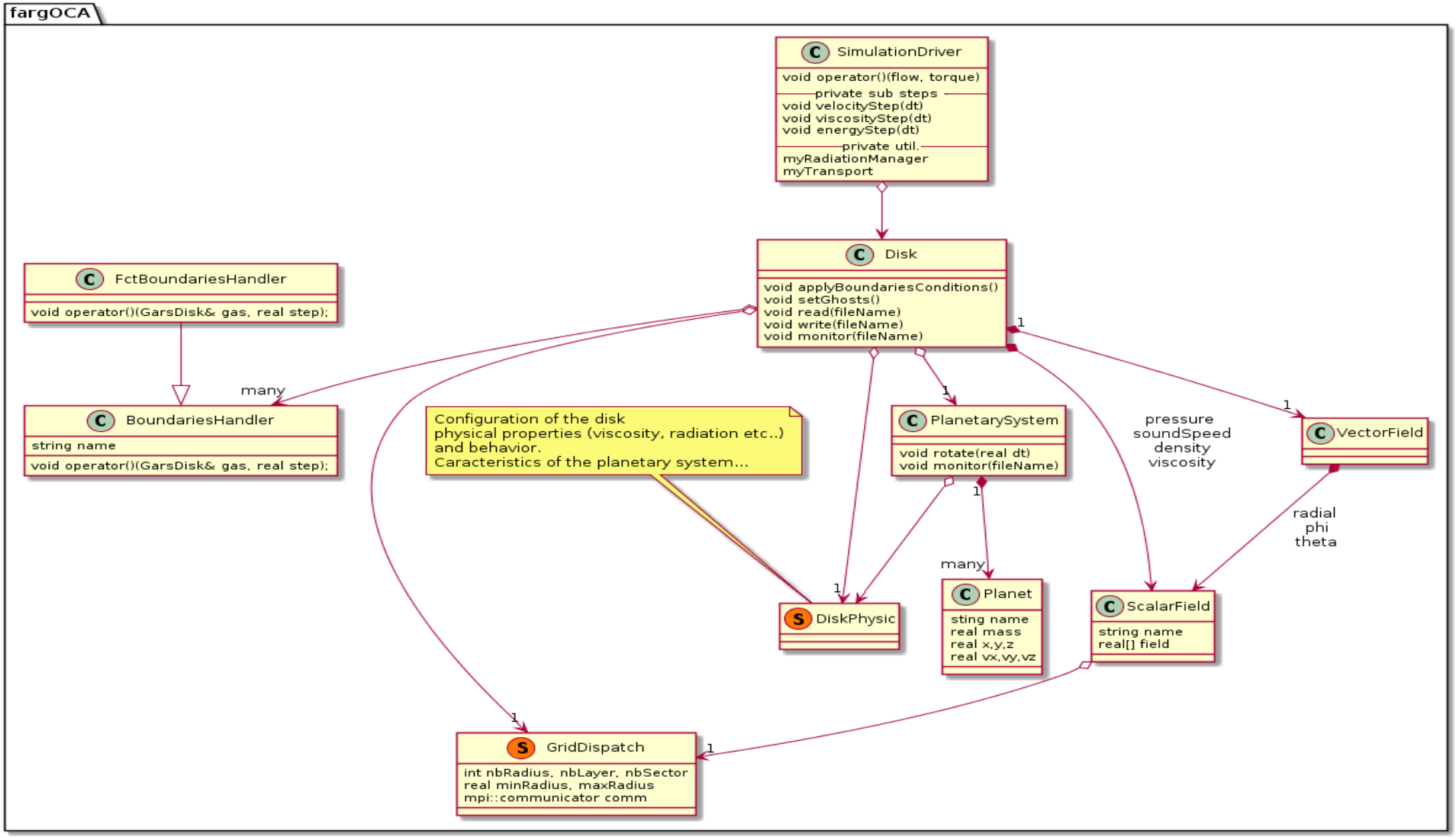
- ✓ You do not "own" your code, your employer does.
  - Good news: your employer probably doesn't care.
- ✓ And so do your retired pro bono contributors and former unpaid interns, unless they signed something (they didn’t, did they ?).
- ✓ We choose GPL.
  - Unless you have industrial partners, you should probably choose that too.
- ✓ You can ignore the law. If you’re lucky, the law will ignore you.

Main Development work

- ✓ Removed all globals: as explained in any software development course, you should not use globals.
- ✓ Merging a 2D and 3D version of the code.

  "copy paste is the last refuge of incompetence" (I. Asimov).
- ✓ Replace in house code with external libraries whenever possible (boost, hdf5...)
- ✓ Use more flexible configuration format(s) with usually up to date documentation.
- ✓ Plus the usual stuff (safety, testing , modularity, encapsulation...)

Class Diagram



Data Format

- ✓Moved to HDF5:
  - Already has dedicated libraries in various languages (including python)
  - Self documented: a skilled python user can investigate the file and guess what's in there (velocity fields are names velocity , density field is named "density" etc...)
- ✓Decided to switch from... files... to a *single* HDF5 file.
- ✓Can't loose/mix up thing
- ✓Contains
  - the data (field values, planetary system state)
  - the physic configuration
  - the time in the simulation

Data Format

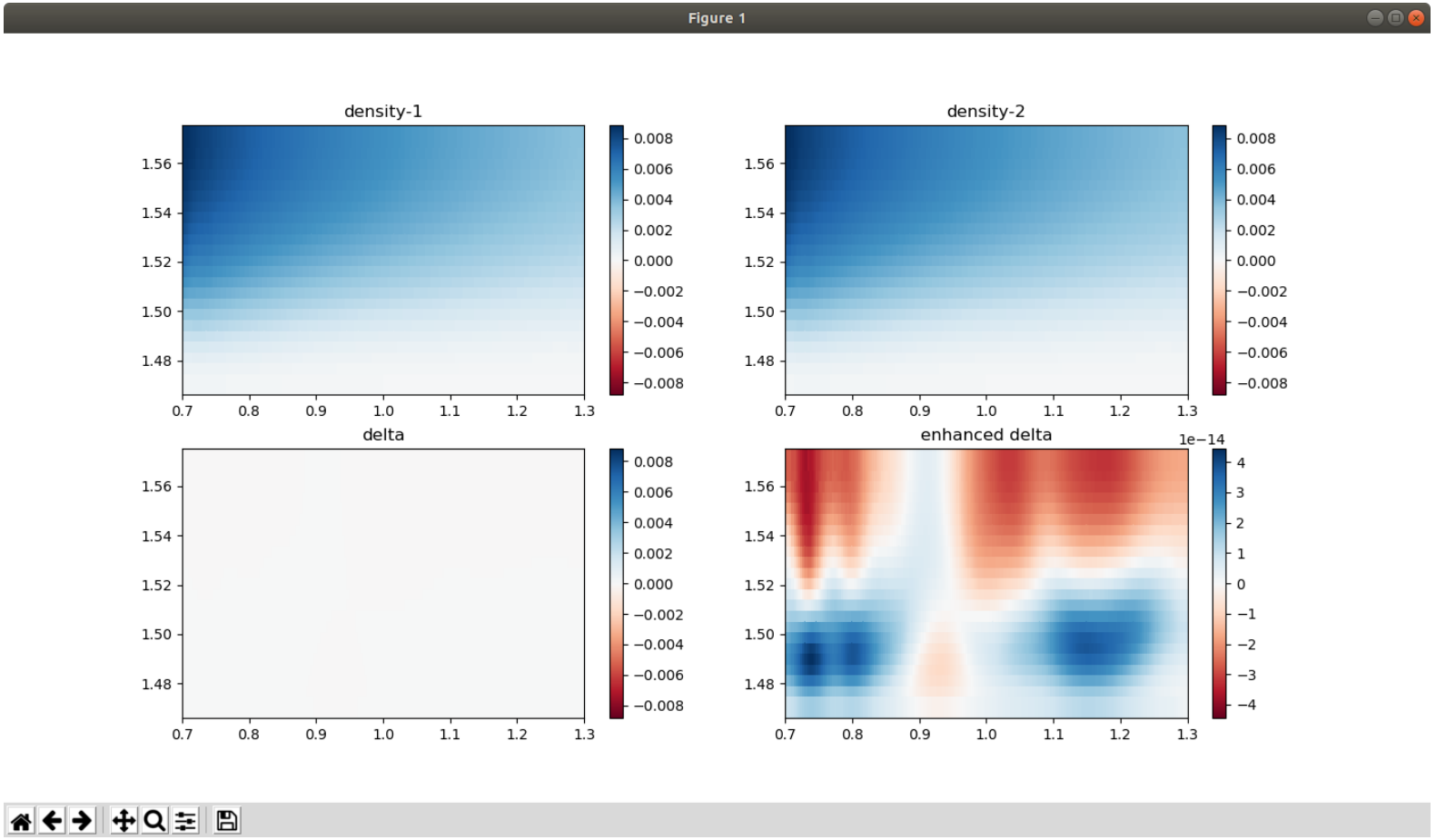


Figure 1
density-1
density-2
delta
enhanced delta
1e−14
1.56
1.54
1.52
1.50
1.48
0.7
0.8
0.9
1.0
1.1
1.2
1.3
0.008
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
−0.008
4
3
2
1
0
−1
−2
−3
−4

How To

(short version)

- ✓ Step 1: Feed a configuration file to the disk initialization tool.
  - – you can toy around with it.
  - – check the fields
- ✓ Step 2: Run the simulator.
  - – Restart is free: the simulator will automagicaly pick up the most recent disk and start from there.
  - – You don't care too much how many nodes are available (more is still better) nor for how long

The Forge

Until you become famous enough to have your code available as linux packages the forge will be you potential users link.

- ✓ Users:
  - Get the code
  - Read the documentation through de wiki.
  - Submit problems through the issue tracker.
- ✓ Contributor:
  - Propose merge requests
  - Discuss changes, regressions and new features through the issue tracker.
- ✓ Integrator:
  - accept merge request, maybe asking for changes and clarifications.
- ✓ For most commits, the test suite is run on the cluster.

After the first release

- ✓Replace non uniform grid with multi grid and then
  - Add cuda support through new OpenMP standard
  - Study the possibility to add explicit platform independent (not a typo) SIMD
- ✓Exploit computation/comunication overlaps.
- ✓Pseudo automatic ghost handling (explicitly done at this point).
- ✓Python interface to the code (ongoing development actually):
  - provide full access to the object model through python.
  - can be used for disk manipulation scripts.
  - make it possible to address the more often needed adaptation through python plugins.
  - Allow for binary distribution.
  - Mostly useless, for sure, but kids loves it.

Observatoire de la CÔTE d'AZUR

Membre de UNIVERSITÉ CÔTE D'AZUR

FARGOCA

Any Question ?