DEVELOPER
DAYS2014
EUROPE

Contributing to Qt – hands on

Giuseppe D'Angelo

KDAB

About me

- Giuseppe D'Angelo
- Qt user since Qt 3
- Joined KDAB in 2012
- Qt Approver
  - Ask me about Core, GUI, OpenGL...
- “peppe” on the interwebs

History

- Historically it has always been cumbersome (if not impossible) for 3rd parties to contribute to Qt
  - No clear contribution model
  - No technical facilities for external contributors

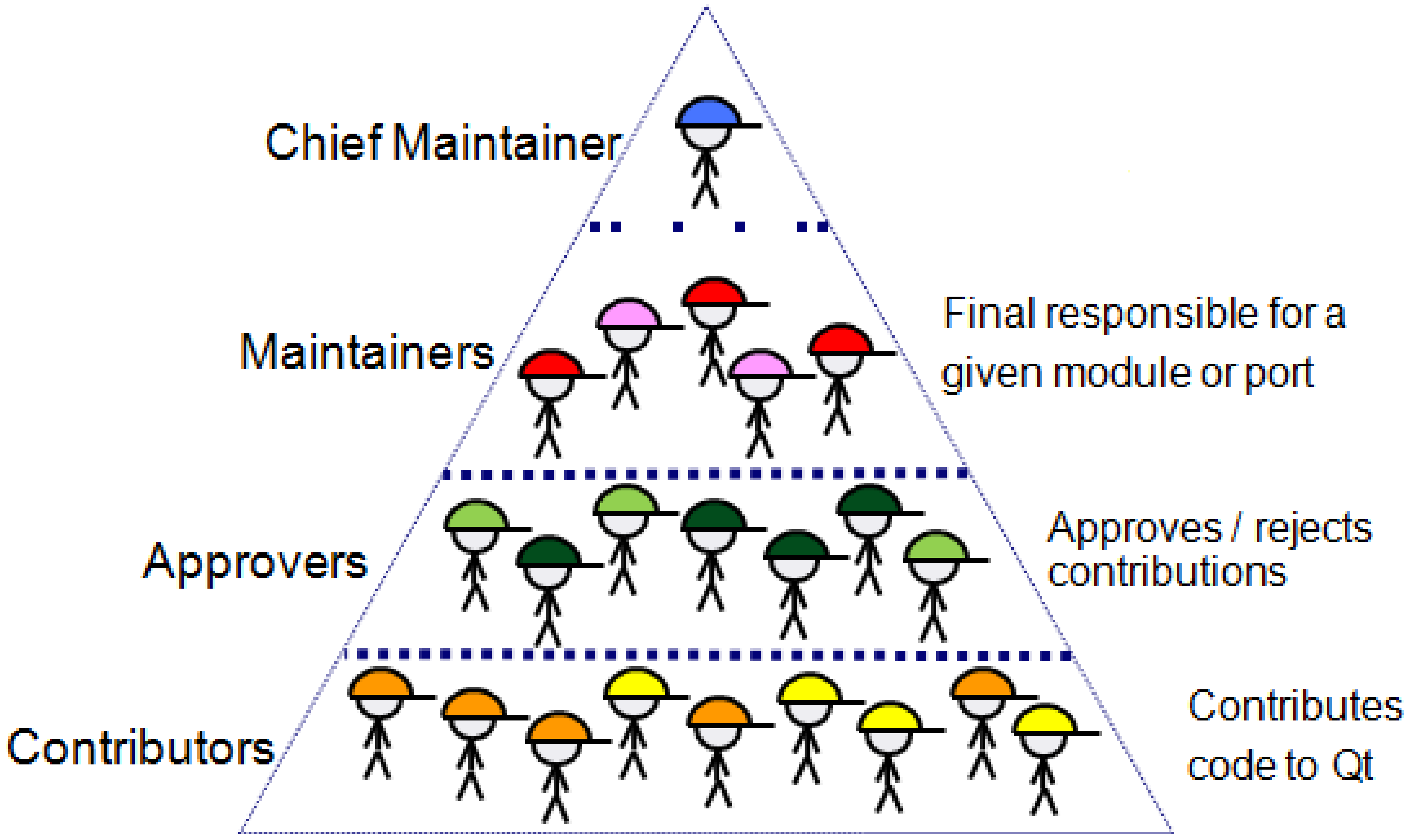
Qt Project

- In 2011 the Qt Project is born
- A Governance Model is put in place to allow external contributions to drive Qt development
  - Meritocratic
  - Inclusive
  - Fair
  - Transparent

Benefits

- Drive Qt development toward one's needs
- Richer, more stable releases
- More opportunities for Qt experts
- Overall: more broad Qt ecosystem

Roles in the Qt Project



Chief Maintainer

Maintainers

Final responsible for a given module or port

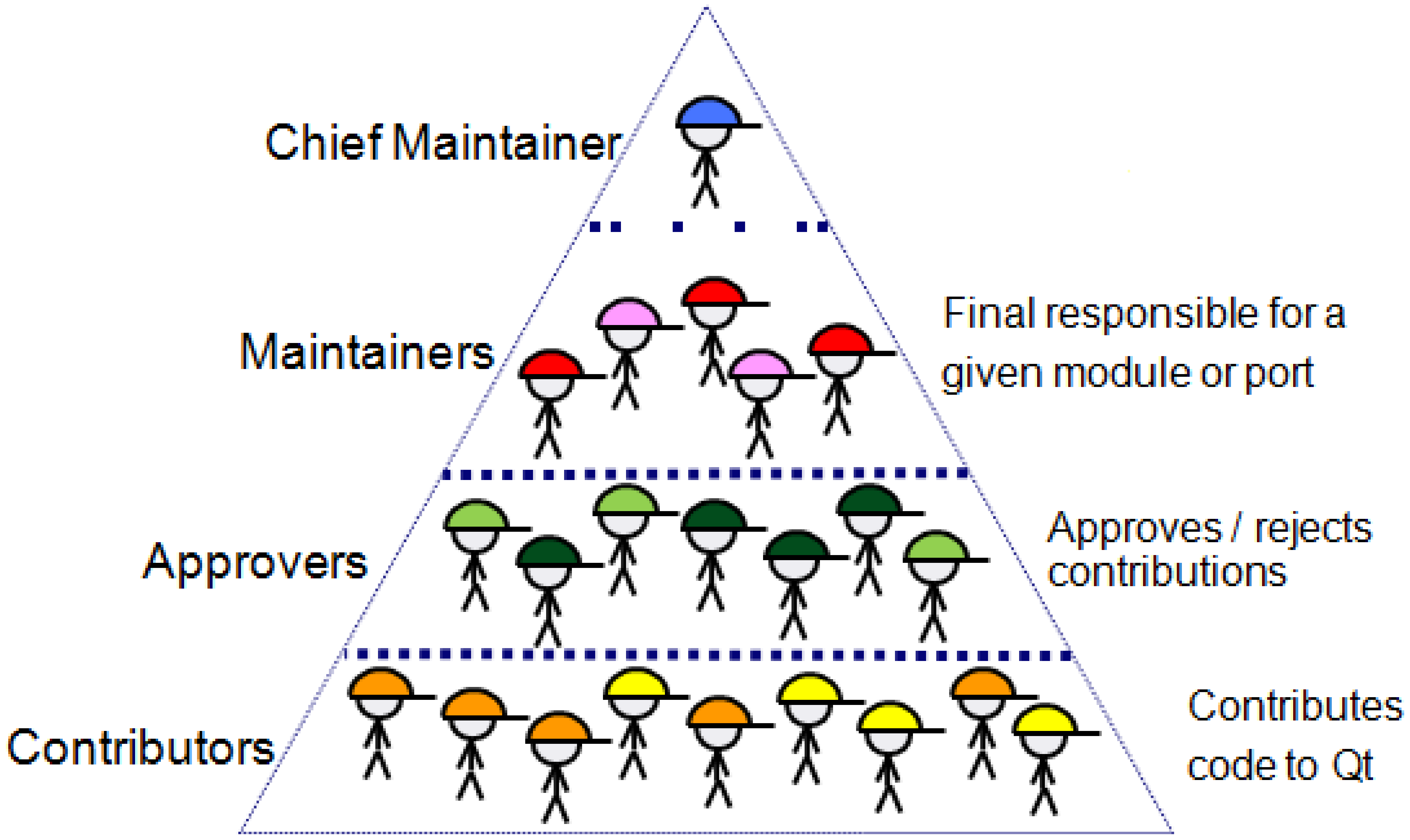
Approvers

Approves / rejects contributions

Contributors

Contributes code to Qt

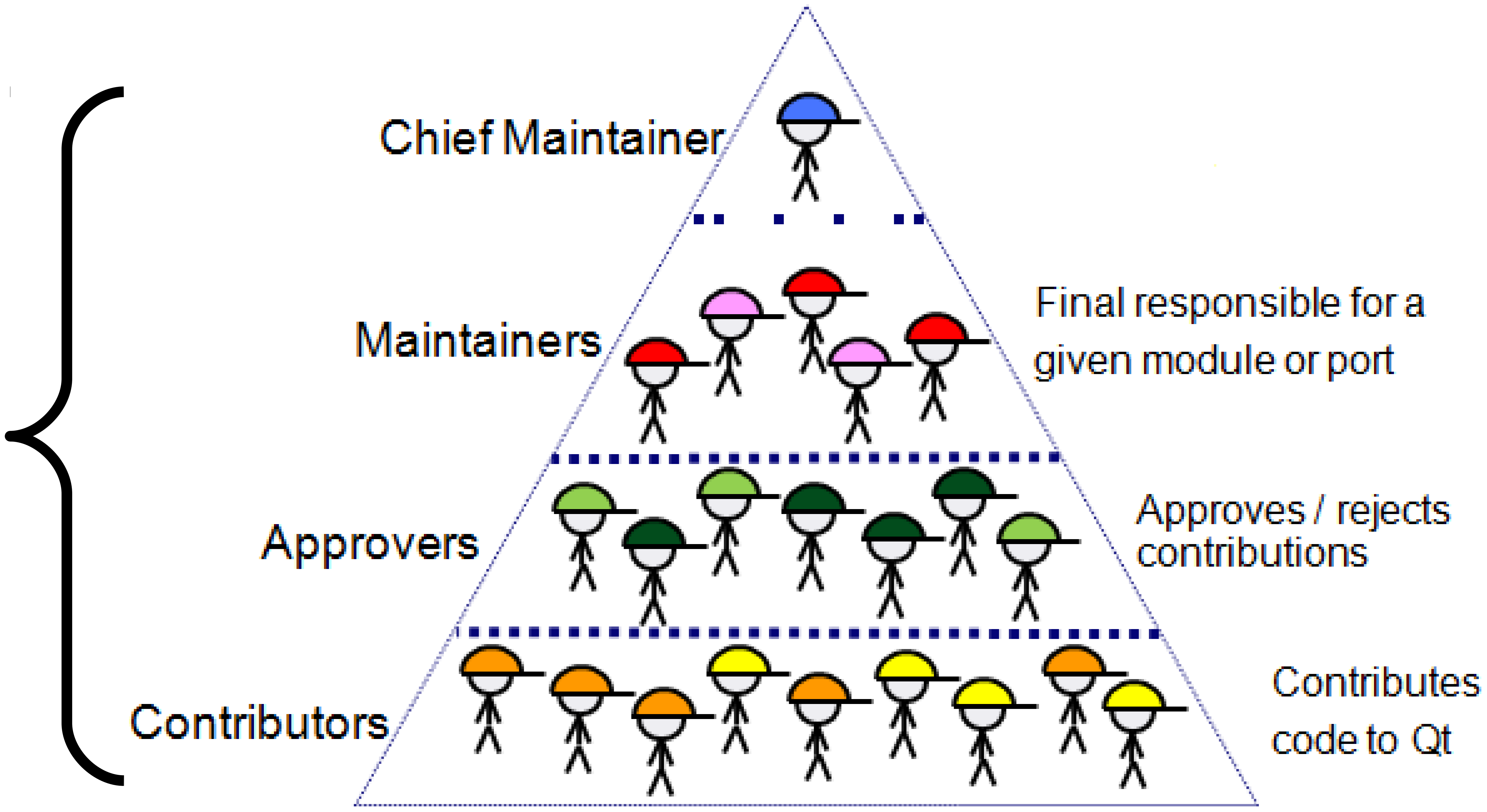
Roles in the Qt Project


Chief Maintainer
Maintainers
Final responsible for a given module or port
Approvers
Approves / rejects contributions
Contributors
Contributes code to Qt

Ordinary users

Roles in the Qt Project

COOL GUYS

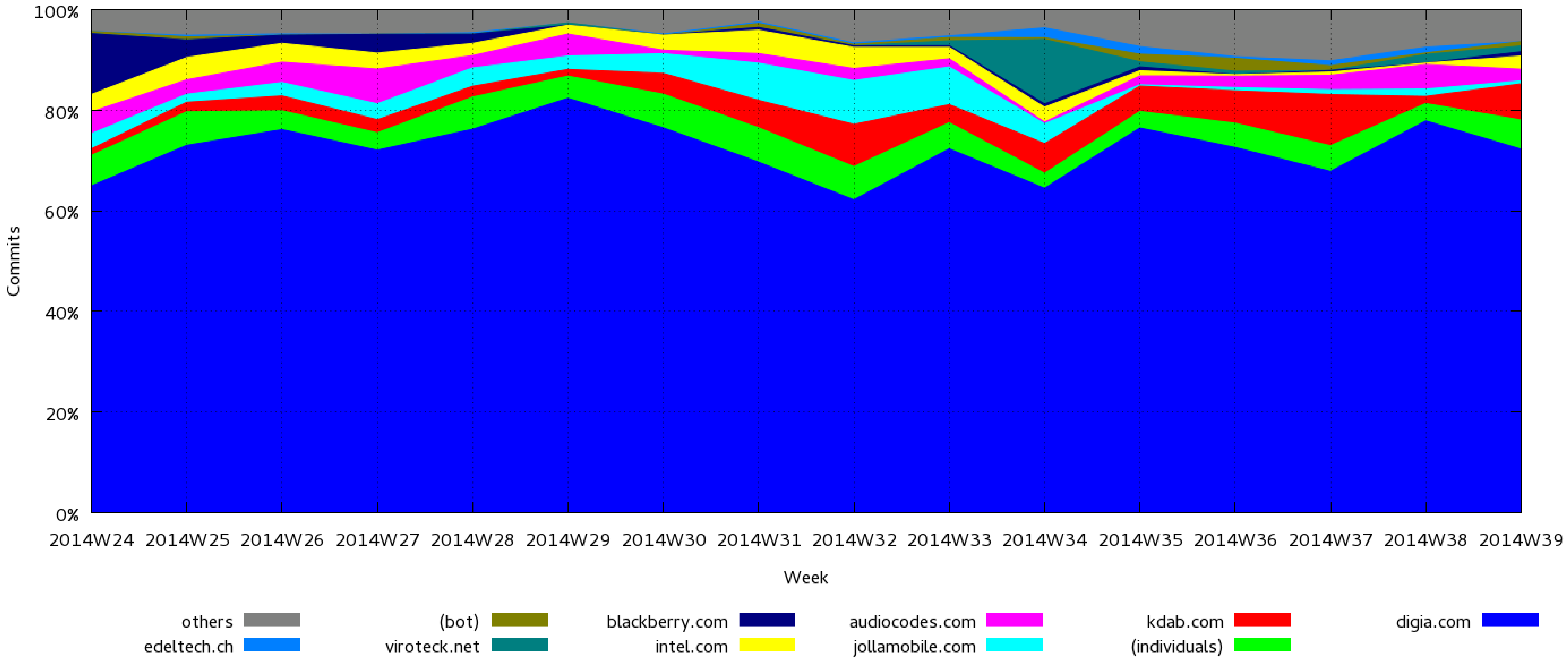

Chief Maintainer
Maintainers
Final responsible for a given module or port
Approvers
Approves / rejects contributions
Contributors
Contributes code to Qt

Ordinary users

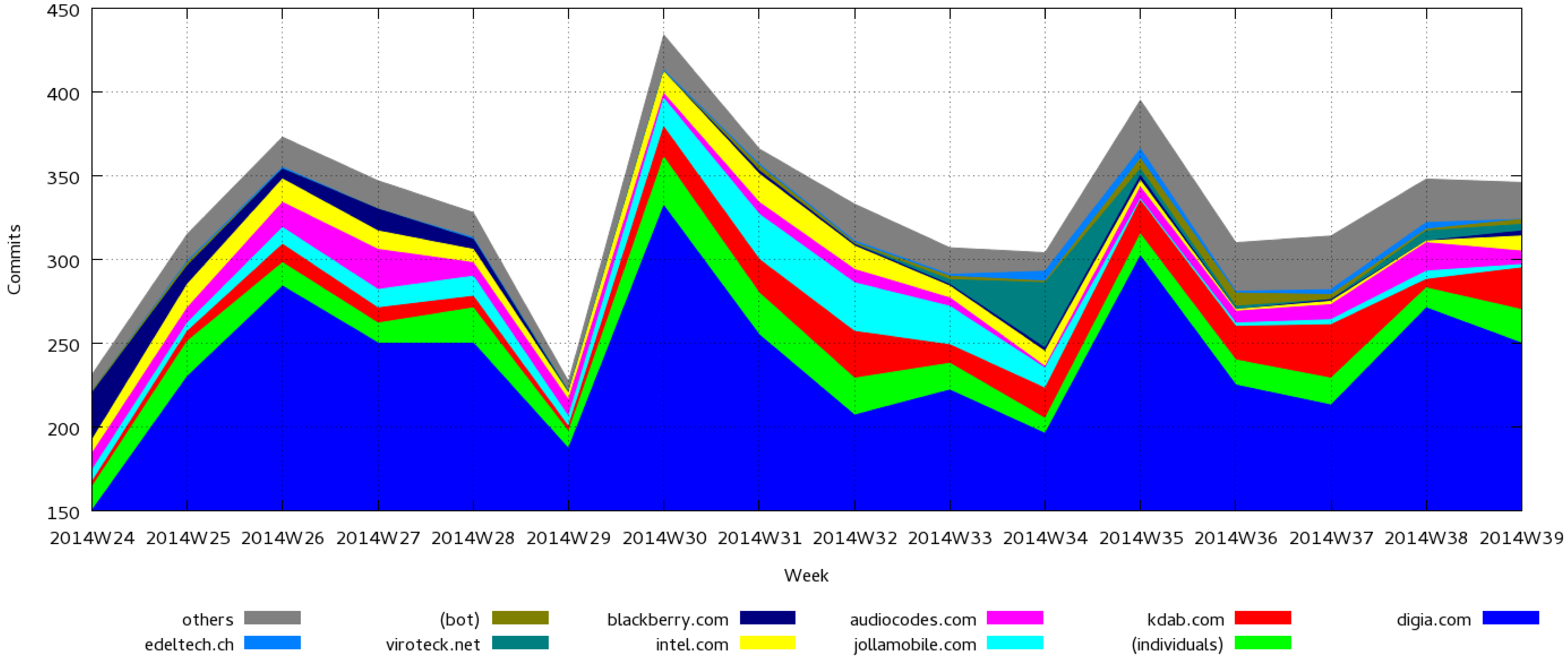
Some numbers

- ~300 commits per week on average
  - ~60 of which (20%) not from Digia

Some numbers: % of commits

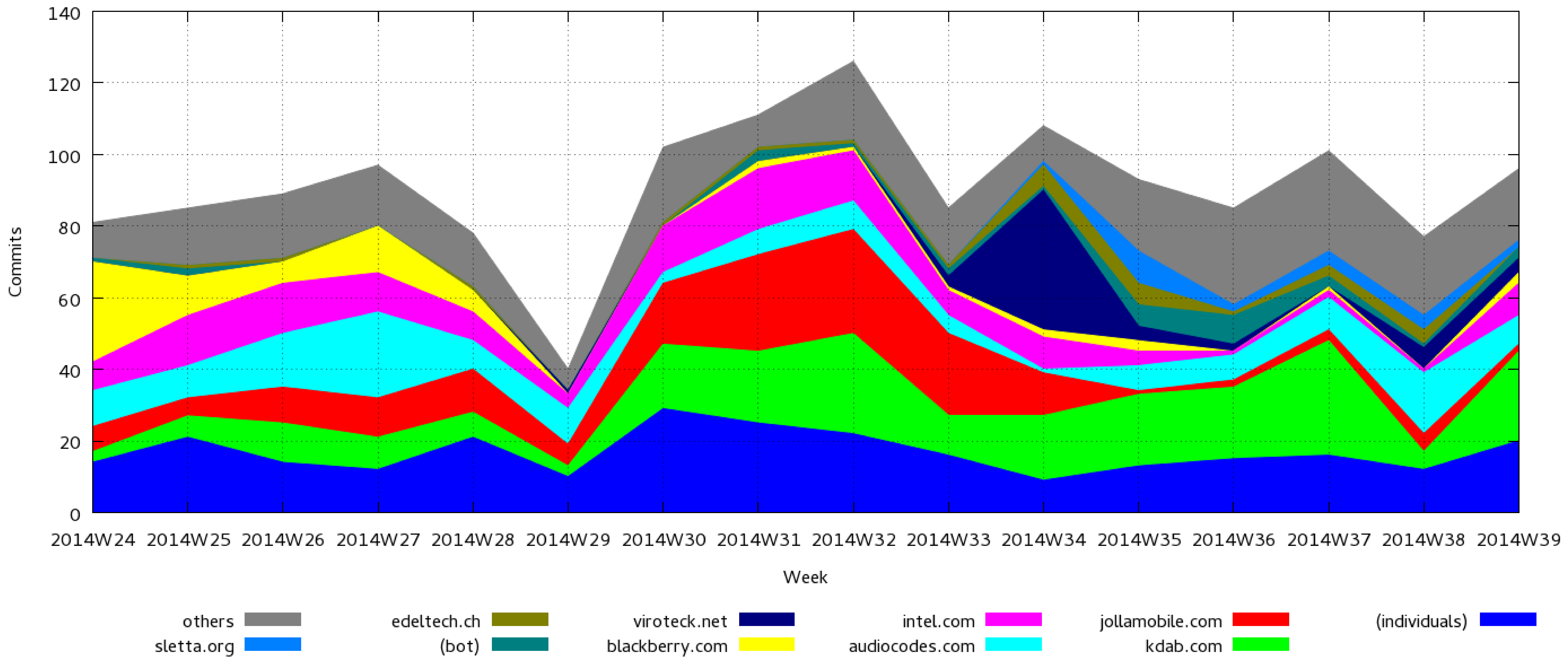


Some numbers: # commits



DEVELOPER DAYS2014 EUROPE

Some numbers: # commits w/o Digia



Let's begin

- One-time preliminary setup
- Make a patch

Setup

- Create a JIRA account *bugreports.qt-project.org*
- Accept the CLA *codereview.qt-project.org*
- Upload your SSH key
  - Optionally: configure SSH for *codereview.qt-project.org*

Contributor License Agreement

- Legal requirement to be able to push any code
- The author **retains copyright**
- Gives Digia authorization to relicense
- Frees Qt Project / Digia from legal responsibilities
- Different agreements available (for individuals or organizations)

Almost there...

- Clone Qt from git
  - either by cloning *qt5.git* and using the *init-repository* script
  - or just clone the modules you need.

    In this case, configure SSH / Gerrit as a remote
- Install the commit-hook
- Build Qt
  - Since we're modifying Qt, pass *-developer-build*

Make a patch

- Which branch to pick?
- Hack, hack, hack
  - Coding style
  - Coding guidelines
- Commit
- Push
- Get feedback, improve, get it merged

Which branch to pick?

- Qt now uses versioned branches
  - 5.3, 5.4, dev
  - When close to a release, 5.x.y created
- Target the **closest** branch (which is still open)
  - Bug fix against 5.3 series => 5.3
  - Destabilizing change => dev
  - New feature => dev
- Use common sense if in doubt (or ask)

A word about Qt 4...

- No new features can go in Qt 4
- Bug fixes can be applied to Qt 4 if and only if
  - they don't apply to Qt 5, or
  - they do apply and they have already been fixed there
- Use the git-qt-cherry-pick script to ease the process

Hack, hack, hack

- Let's make a patch!

Coding style & guidelines

- They're extensively documented on the wiki
- Long list of “dos and don'ts”
  - Code style
  - API design
  - Commit messages
  - General rules about the project
- Again, privilege common sense

Commit!

- It's git => it has no specific workflow
- Easiest one: create a feature branch and commit in it
  - `git checkout -b featurebranch`
  - `# hack hack hack`
  - `git add file1 file2`
  - `git commit`
- The commit-hook will automatically insert a Change-id below your commit message (leave it alone, it's for Gerrit)

Push the patch for code review

```
git push gerrit HEAD:refs/for/targetbranch
```

Getting the patch reviewed

Getting the patch reviewed

- Find suitable people to review (and approve) the patch
- git log, git blame
- Add the maintainer(s) of the module
- Ping people around on IRC / mailing list / JIRA

Getting the patch reviewed

- Scoring system used
  - +2 “approved”
  - +1 “ok for me, but someone else must approve”
  - 0
  - -1 “not ok as-is, but if you improve it I'll accept it”
  - -2 “totally disagree with the feature or the given implementation”
- Scores don't sum up

Oh no! Someone -1'd my patch

- Don't worry! Happens all the time
- Extremely likely the same people also left comments about what to improve (and how to do it)
  - E.g. “style violation” => read the style guide
  - “what happens if someone does this-and-that...”
  - “missing docs”

Improve the patch

- Learn from the feedback
  - Don't be afraid to ask for more information!
- Go back to the code
- Fix the issues
- Amend the commit and repush it
  - `git add file1 file2`
  - `git commit --amend`
  - `git push gerrit HEAD:refs/for/targetbranch`
- Do **not** create a new commit

Docs, tests

- Docs are mandatory for any new feature
- Tests are almost mandatory for any new feature or bugfix
  - Exceptions to be made on a case-by-case
- Qt uses QTestLib for its own tests
  - Each module has tests in `tests/auto/module/subdir/class`

My patch got approved! What now?

- Press the “Stage” button on Gerrit and cross your fingers
  - Staging isn't automatic
  - Prevents problems in case of series of patches with dependencies
- The patch is now in the hands of the CI system

Continuous Integration

- All patches landing in Qt must pass CI
  - They must build on all CI configurations (currently ~30)
  - On several of them, all the enforced tests must all pass
- Unfortunately a bunch of staged patches are tested all together
  - If any of them causes a failure, all of them will not pass
  - If you're positive your patch didn't cause the failure, stage again
  - Otherwise, *fix the code and push again*

Success!

- The patch got merged

Too complicated?

- If you suspect you have found a bug in Qt...
- And you don't want to investigate how to fix it...
- And you don't want to go through the whole contribution process...

Got a Qt bug?

KDAB can fix it for you

fixmyqtbug.com

Thank you!

- Other useful resources:
  - **qt-project.org/contribute**
    - includes all these instructions, guides, troubleshooting…
  - *development* mailing list
  - #qt-labs @ irc.freenode.net

Questions?

Thank you!

- Other useful resources:
  - qt-project.org wiki
    - includes all these instructions, guides, troubleshooting…
  - *development* mailing list
  - #qt-labs @ irc.freenode.org