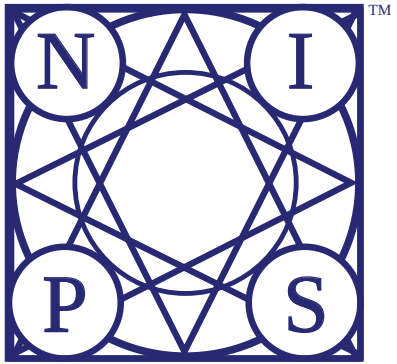
German Research Center for Artificial Intelligence

The Eigen3 Matrix Library

Machine Learning Open Source Software 2018: Sustainable communities

Dr.-Ing. Christoph Hertzberg
DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Dr. h.c. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de



Outline

- Eigen a C++ linear algebra library
- History and User/Developer Community
- Assessments

Slides are based on Gaël's Talk at B-Boost (https://b-boost.fr/)

Matrix computation is everywhere

- Various applications:
  - simulations, simulators, video games, audio/image processing, design, robotic, computer vision, augmented reality, **Machine Learning**, etc.
- Various numerical tools:
  - numerical data manipulation, space transformations
  - inverse problems, PDE, spectral analysis
- All want performance:
  - on standard PC, smartphone, embedded systems, etc.
  - real-time applications

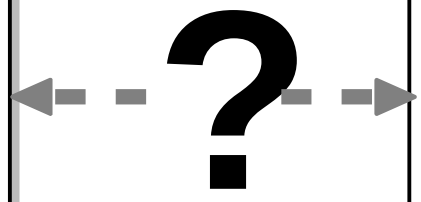
Matrix computation?

German Research Center for Artificial Intelligence

MatLab & co

+ friendly API
+ large set of features
- math only
- extremely slow for small objects

→ **Prototyping**

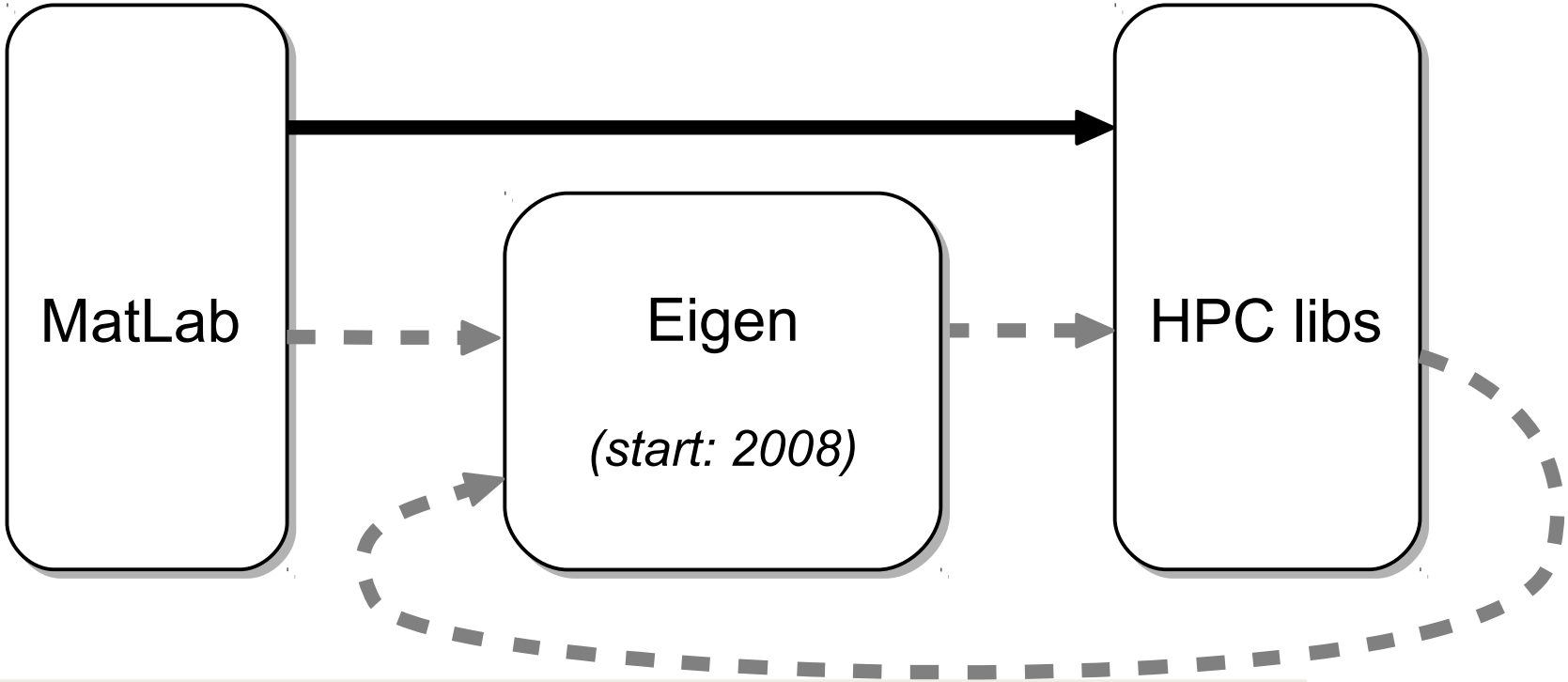


HPC libs

+ highly optimized
- 1 feature = 1 lib
+/- tailored for advanced user / clusters
- slow for small objects

→ **Advanced usages**

→ take best of two worlds?



Some key features ↔ Design rules

- Pure C++ template library
  - header-only (no config step, no build step), no dependency
  -

```
$ hg clone https://bitbucket.org/eigen/eigen
```

```
#include <Eigen/Eigen>
using namespace Eigen;
int main() {
  Matrix4f A = Matrix4f::Random();
  std::cout << A.transpose()*A << std::endl;
}
```

```
$ g++ -I eigen -O2 example.cpp -o example
```

Some key features ↔ Design rules

- Pure C++ template library
  - header-only (no config step, no build step), no dependency
  - opensource (MPL2), stable API

  → easy to install & distribute
- Multi-whatever
  - OS: Linux, Windows, OSX, Android, IOS, etc.
  - Comp: GCC, Clang, MSVC, ICC (Intel), XLC (IBM)
  - HW: SSE/AVX/Xeon-Phi, ARM/NEON, Altivec/VSX/Zvector, MSA, GPU

  → raspberry-pi ↔ cluster nodes

Some key features ↔ Design rules

- Generic & versatile
  - tiny matrices → large & dense → large & sparse
  - matrix manipulation → matrix products → solvers
  - custom scalar types

  → “unified API” - “all-in-one”
- Optimized at all scales
  - explicit SIMD, cache-aware kernels, multi-threading
  - lasy evaluation, fused operations, expression rewriting, etc.

  → no compromise on speed

Summary

- Goal = ideal compromise between
  - versatility
  - ease of use
  - performance

History

&

Open-source Community

Genesis

- Sept. 2007: start of Eigen2
  - proof-of-concept initiated by Benoit Jacob
  - part of KDE (!)
  - open repository
  - open discussions on mailing list/IRC/bugtracker
    - API/internal design, environment, license, etc.

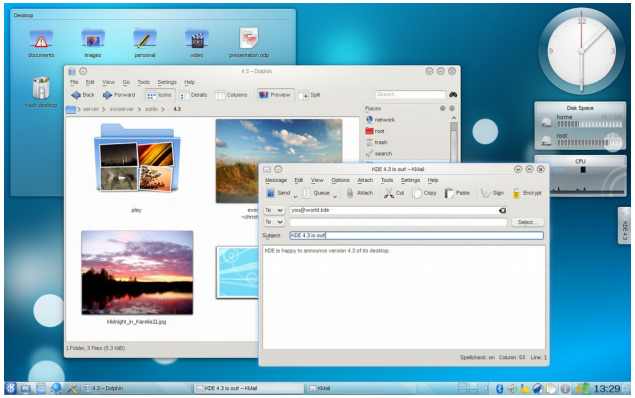
→ truly open development

Part of KDE?

- Pros
  - infrastructure
  - initial community
  - open development approach
  - **packaged by all Linux distributions**

    \+ macport, homebrew, cygwin, etc.
- Cons
  - out of scope

    → we rapidly moved away



Truly open development

- Pros
  - get users involved
    - including myself!
  - diversity
    - blend different skills, backgrounds, application domains

  → reach high quality API
- Cons
  - reaching consensus takes time…

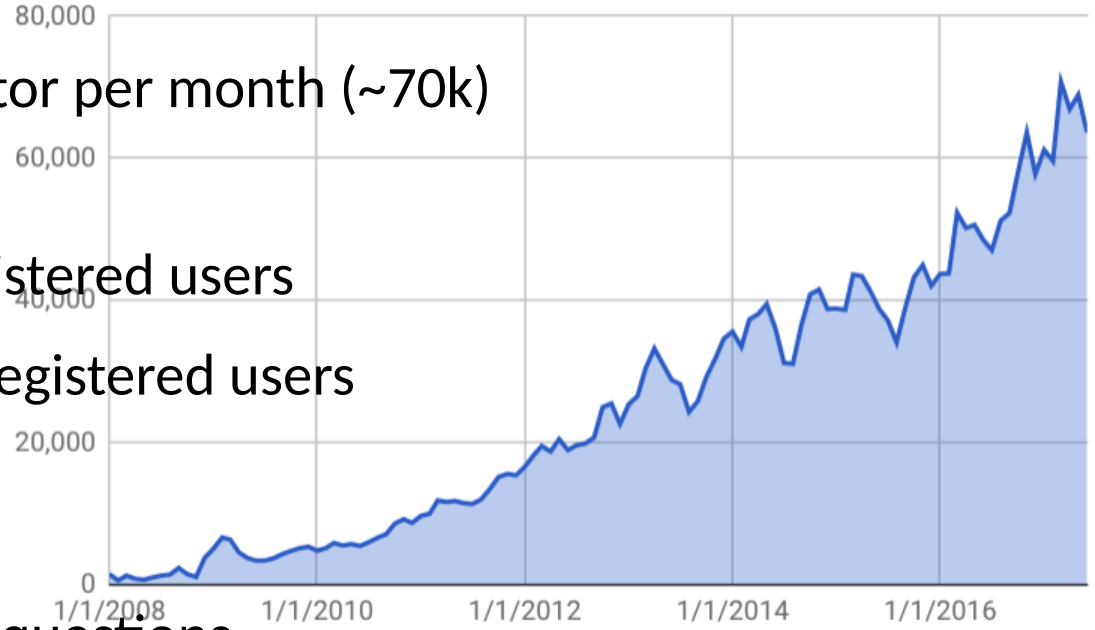
… sometimes years!

Users

Who? How Many?

Large pool of users

- Numbers
  - Web site: unique visitor per month (~70k)
  - Bugzilla: 1058 registered users
  - Mailing List: 702 registered users
  - Forum: 2865 topics
  - StackOverflow: 2400 questions



Users: examples

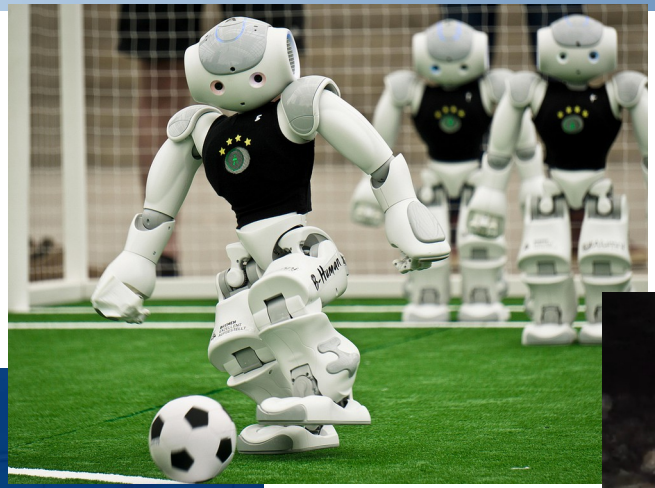
- Researchers: vision, graphics, robotics, physicists
  - CEA, EDF, “The ATLAS experiment” at the LHC (CERN)
- Inria: used by numerous teams and software
  - CGAL, OpenVibe, SOFA, SciLab, …
- Open-source projects
  - ROS, PCL, MeshLab, Blender plugins, Cufflinks, …
- Toolboxes:
  - RcppEigen, GDL, MP Matlab Toolbox, Scilab, …
- Companies
  - Many startups & new projects

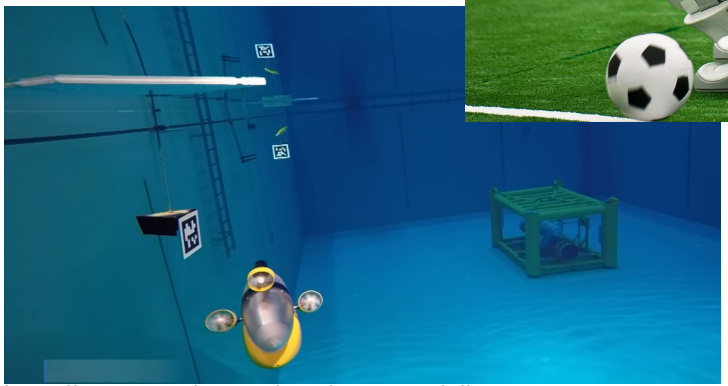
Users: Google

- Computer vision (2008→)
  - e.g., StreetView/Ceres
- Machine Learning (2013→)
  - SSE → AVX → $100 million saving cost!
  - TensorFlow

Users: DFKI

German Research Center for Artificial Intelligence





https://www.youtube.com/watch?v=ngqAfPib40s

https://www.youtube.com/watch?v=etB5TUkBFXc

Many users: the pro and cons

- Many users: Yay!
  - but…
    - many questions
      - ~~ML/Forum~~
      - Stack-overflow (since 2011) → users really help each others!
    - many feature requests
    - many “bug” reports
- A hope: our users are developers …

Attracting contributors?

How does it work in practice?

→ history of main contributors

It all started with academics

- Benoit Jacob (2008→2011) / during PhD,postdoc
  - for “fun” → hired by Mozilla then Google
- Gaël (2008→?) / post-doc, researcher
  - started as both user and main co-maintainer
- Hauke Heibel (2009→2011) / during PhD
  - user → MSVC fixes → API/internal design, features
- Jitse Niesen (2010→2015) / assistant prof.
  - doc → features, general bug fixes, API/internal design
- Christoph Hertzberg (2013→?) / during PhD/postdoc
  - user → user support → bug fixes, review, API/internal design

→ **key: truly open development**

Freelancers

- Thomas Capricelli (2008→2011)
  - ~~1 non-linear solver~~, infrastructure
- Konstantinos Margaritis
  - ARM/NEON, PowerPC (Altivec/VSX,ZVector)
  - paid via bounties from IBM

Eigen @ Inria

German Research Center for Artificial Intelligence

- Contributors
  - Gaël (2009→?) / researcher
  - Désiré Nuentsa (2012→2013) / full-time engineer
- Infrastructure
  - CI/SonarCube
  - Server, web hosting, access to advanced CPUs

Eigen @ Google

- Benoit Steiner (2014→2018)
  - creator of the Tensor module, AVX/AVX512/CUDA
- Rasmus Larsen & Eugene Brevdo (2016→?)

- Write access to the repository
- Take part of strategic discussions
  - e.g., license (LGPLv3 → MPL2), minimal requirements
  -
- Challenges
  - Find compromise between our ideals and their needs

Some other industrial contributors

- Intel: code for MKL compatibility (2011→)
- CodePlay: code for SYCL support (2016→)
- AMD: code for HIP support (2018)
- Wave Computing: support for MSA (2018)

→ Nearly no features from industrial users :(

- they keep them for them!
- no patience to iterate to reach our quality standards
  - API, genericity, header-only, multi-platforms, maintainability, unit-testing, doc, …

Integration of new features

- We guarantee API stability
  - takes time to reach

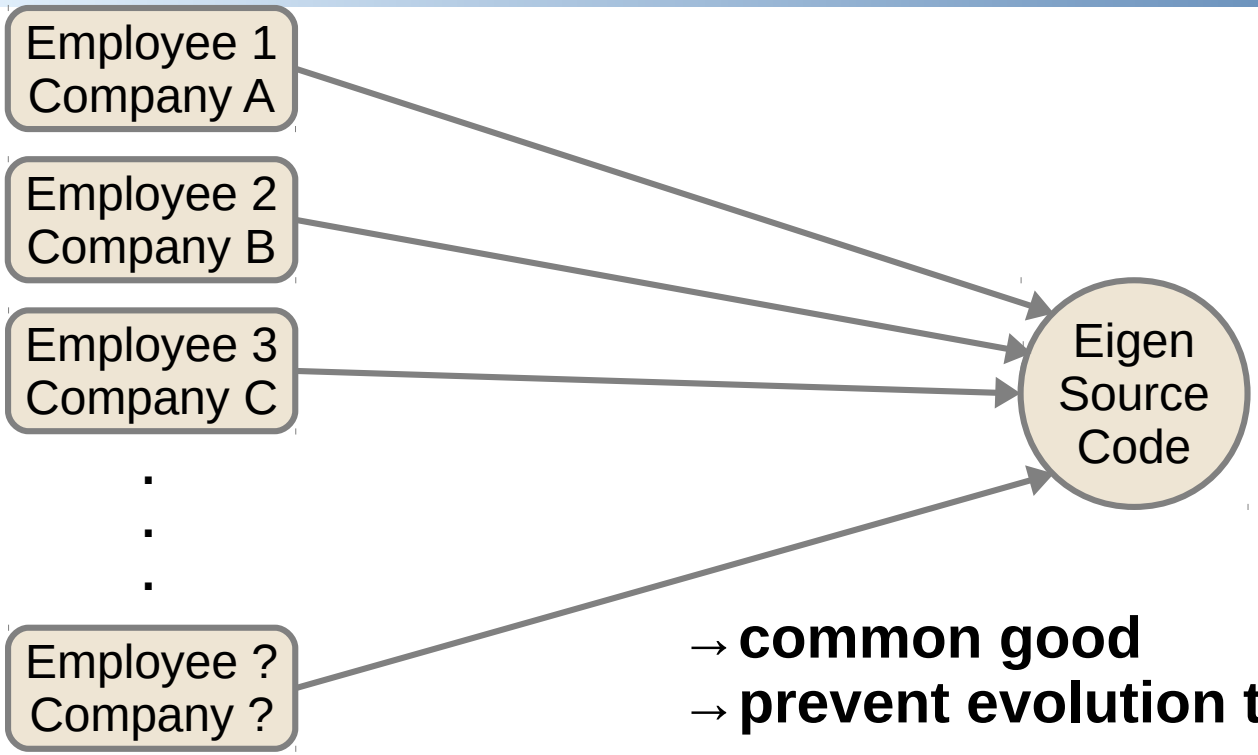
→ Set of “unsupported” modules for quicker adoption and maturation

- ex. Non-Linear-Optimizers
- ex. Google's Tensor module

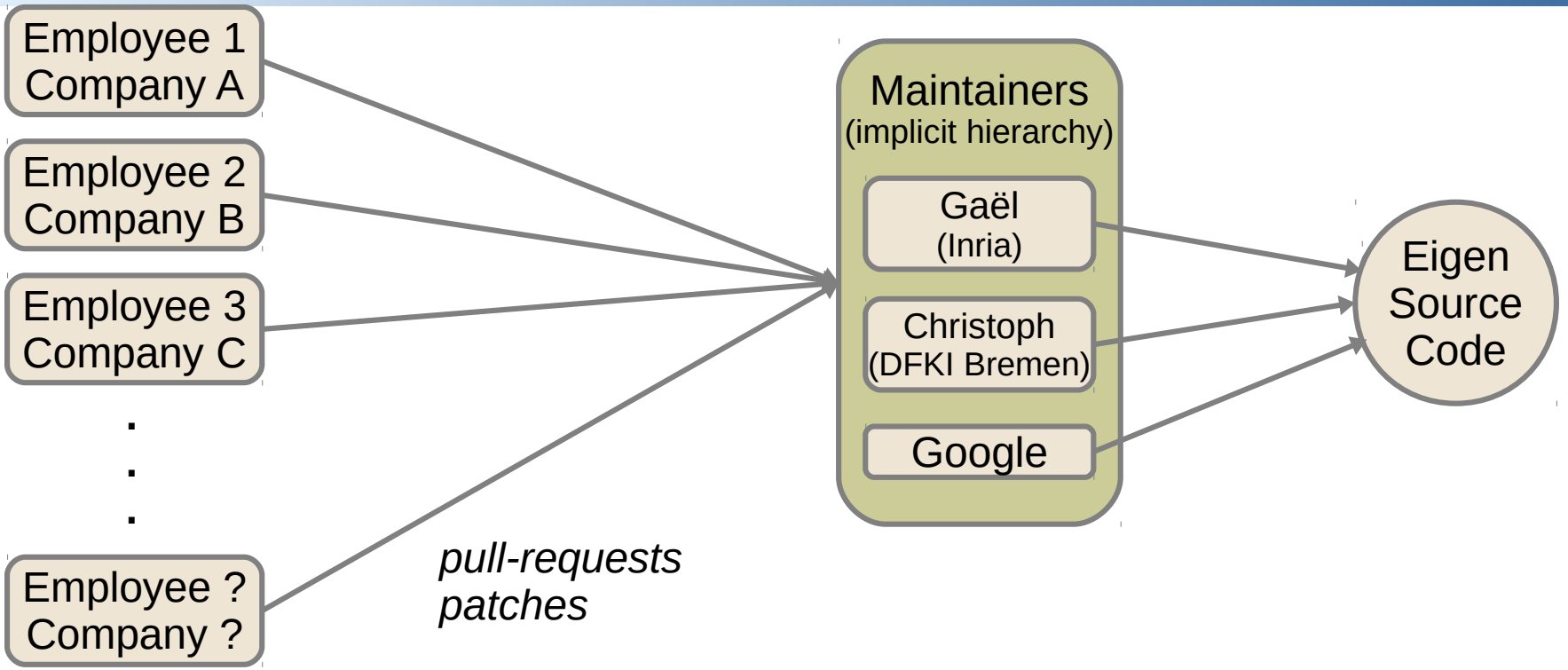
… and numerous occasional contributors

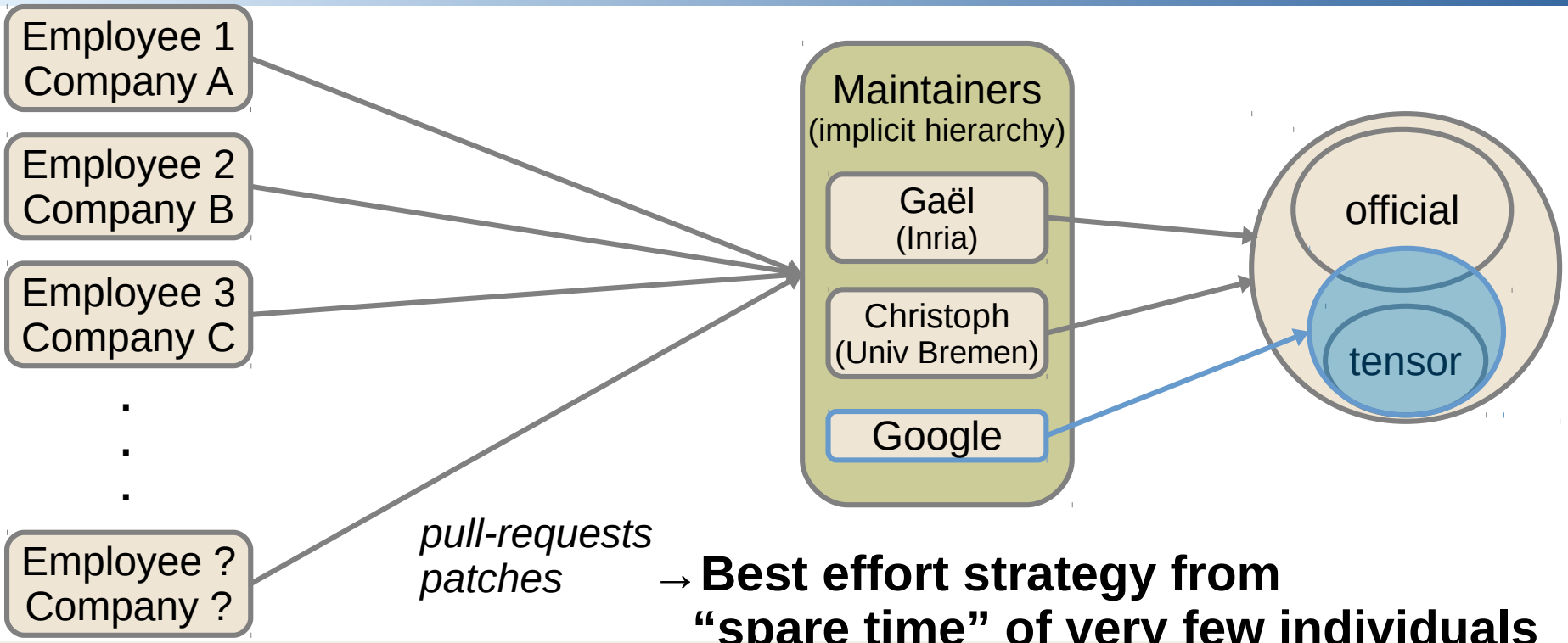
- Total of about 240 contributors to the code
  - from small features to bug fix patches
- Plus
  - users helping on API design (~100?)
  - users doing CI/nightly builds
    - Microsoft: included Eigen's tests in their MSVC set of tests

Development model



→ **common good**
→ **prevent evolution to proprietary code**


Employee 1
Company A
Employee 2
Company B
Employee 3
Company C
Employee ?
Company ?
pull-requests
patches
Maintainers
(implicit hierarchy)
Gaël
(Inria)
Christoph
(DFKI Bremen)
Google
Eigen
Source
Code



→ **Best effort strategy from "spare time" of very few individuals**

Limitations

- Few maintainers for many tasks
  - Handle pull-requests/patches
    → reviews and (many) iterations/discussions
  - Handle bug report/fixes
  - Coding: new features, improvements, finalize feature proposals, …
  - Unit testing/CI (combinatorial explosion, servers are always broken)
  - Animate the community
    - Help users, organize meetings (3 up to now)
    - Assists volunteers, try to motivate them, … hard in practice

→ needs more “maintainers”

Questions ?