TANGO

Connecting things together

1st Tango Kernel Webinar
cppTango Overview

Reynald Bourtembourg - ESRF

Thomas Braun - ( ) byte physics

Michal Liszcz - S2INNOVATION

https://www.tango-controls.org

Outline

1. **Introduction** (Reynald)
2. **cppTango repository overview** (Reynald)
3. **cppTango dependencies** (Reynald)
4. **How to compile cppTango?** (Thomas)
5. **How to run the tests?** (Thomas)
6. **How to add a new test?** (Thomas)
7. **Architecture overview** (Michal)
8. **Practical example: Code navigation. What happens when an attribute is read?** (Michal)
9. **Questions** (You)

Introduction

http://gph.is/28RgAAs

Goals:

- ✓ Share knowledge on Tango Kernel
- ✓ … before the Tango Creators retire or get affected by Alzheimer or MIB
- ✓ Encourage contributions to Tango Kernel
- ✓ ~ 1 webinar per month
- ✓ Questions/Answers session at the end

cppTango Repository Overview

https://github.com/tango-controls/cppTango

Development branches

tango-9-lts	9.3-backports
Future cppTango 9.4	(9.3.x development branch)
Requires C++14 at least	Does not require C++11 (Can be compiled on old compilers but might need a more recent CMake version)
Not binary compatible with cppTango 9.3.x	Binary compatible with cppTango 9.3.x
Travis CI: • Latest LLVM (11.0) • Latest GCC (10.2.0) • Ubuntu 20.04 • Debian 8, 9 , and 10	Travis CI: • Debian 7, 8, 9 , and 10
Appveyor: • win32 msvc14 and msvc 15 • x64 msvc14 and msvc15	Appveyor: • win32 msvc9, msvc10, msvc12, msvc14 and msvc 15 • x64 msvc9, msvc10, msvc12, msvc14 and msvc 15

Original slide by Michal Liszcz

Tango 9 LTS

- Future cppTango 9.4 release
  - Breaks ABI (i.e. not binary compatible with 9.3)
- Developed on tango-9-lts branch
  - Since Mar 29, 2019 (9.3-backports branchout)
- Improvements in all areas
  - Bugfixes and features
  - Software quality and safety
  - Tooling and CI infrastructure

Original slide by Michal Liszcz

Tango 9 LTS - Bugfixes

- Solutions may differ from 9.3.x ones
  - No ABI restrictions
  - We can follow the “boy scout rule” and refactor
- Some bugs are resolved on 9.3-backports first and wait for forward port

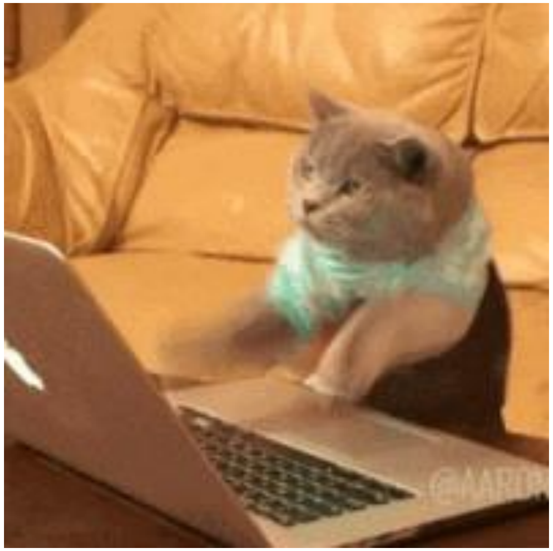
http://gph.is/1JIu1fj

From an original slide by Michal Liszcz

Contribution Rules

Advices in Contributing.md:

- **Discuss addition and changes first** => Github issue



http://gph.is/2vIv8f0

http://gph.is/2bVPjt4

Contribution Steps

- ✓ Fork the repository to your own user
- ✓ Add your fork as new remote:

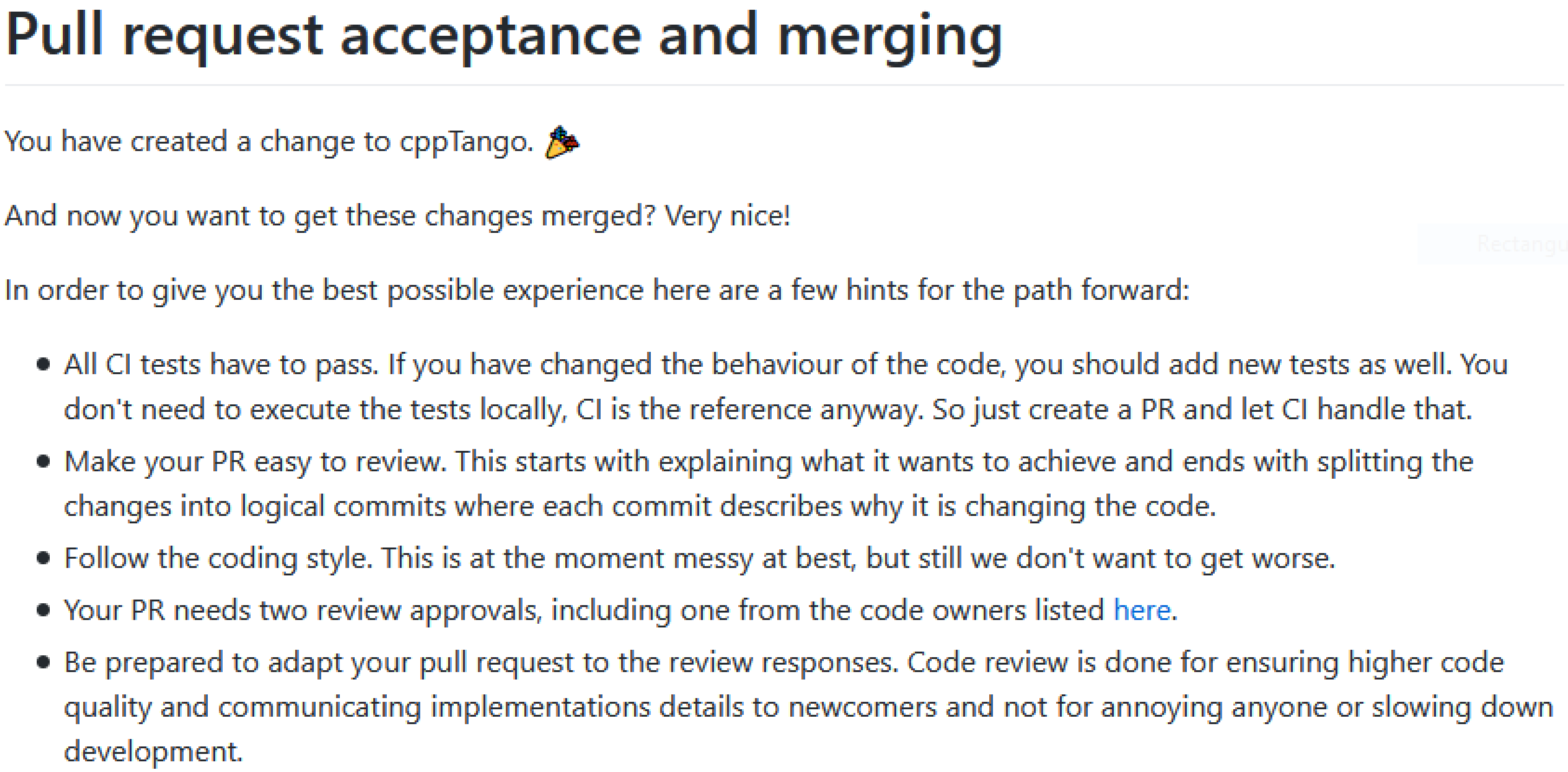
```
git remote add myFork git@github.com:<user>/cppTango.git
```

- ✓ Create a new branch for your work
- ✓ Start hacking
- ✓ Create a pull request with your changes

Your fixes should *always* be based on the default branch `tango-9-lts`.
Only after accepting a PR against that branch, we can start integrating a fix for the current stable version in the `9.3-backports` branch.

Contribution Rules

Extract from Contributing.md:



Pull request acceptance and merging

You have created a change to cppTango. 🎉

And now you want to get these changes merged? Very nice!

In order to give you the best possible experience here are a few hints for the path forward:

- All CI tests have to pass. If you have changed the behaviour of the code, you should add new tests as well. You don't need to execute the tests locally, CI is the reference anyway. So just create a PR and let CI handle that.
- Make your PR easy to review. This starts with explaining what it wants to achieve and ends with splitting the changes into logical commits where each commit describes why it is changing the code.
- Follow the coding style. This is at the moment messy at best, but still we don't want to get worse.
- Your PR needs two review approvals, including one from the code owners listed here.
- Be prepared to adapt your pull request to the review responses. Code review is done for ensuring higher code quality and communicating implementations details to newcomers and not for annoying anyone or slowing down development.

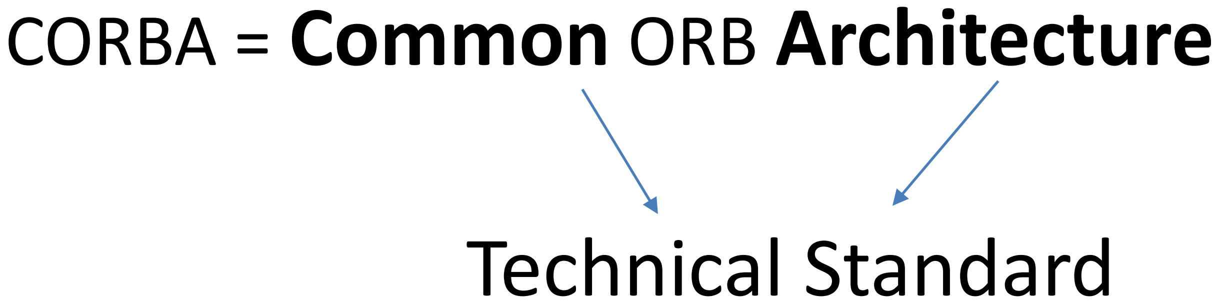
cppTango Dependencies

- omniORB >= 4.2.2 (C++ CORBA)
- libzmq >= 4.0.5 (events)
- cppzmq (zeromq C++ wrapper)
- tango-idl (Tango CORBA Interface)
- cmake >= 3.7
- C++14 compliant compiler (GCC, Clang, Visual Studio >=2015)

cppTango Dependencies: omniORB

CORBA = Common ORB Architecture

cppTango Dependencies: omniORB



CORBA = **Common** ORB **Architecture**

Technical Standard

cppTango Dependencies: omniORB

CORBA = **Common** ORB **Architecture**

Technical Standard

CORBA = technical standard for something called **ORB**

cppTango Dependencies: omniORB

- ORB = Object Request Broker

- ORB = Object-oriented version of RPC (Remote Procedure Call)

- ORB = Mechanism for invoking operations on an object (calling a *Procedure*) in a different (*Remote*) process that may be running on the same or different computer.

- At programming level*,* *remote* calls look similar to *local* calls

cppTango Dependencies: IDL

IDL = Interface Definition Language

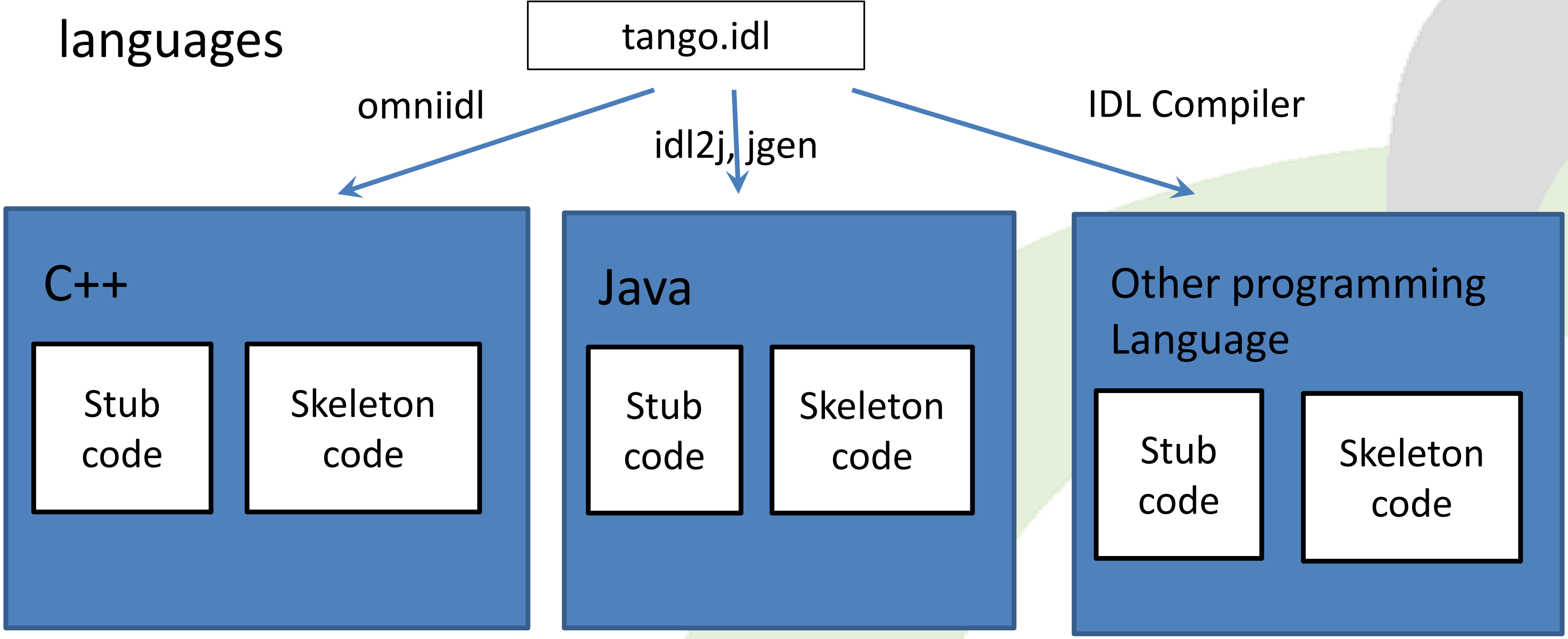
An IDL file defines the public API that is exposed by objects in a server application

When using Tango, there is only 1 IDL file
(usually hidden to the device server and client programmer):
https://github.com/tango-controls/tango-idl/blob/tango-9-lts/tango.idl

IDL = API defined in a way which is independent of any particular programming language

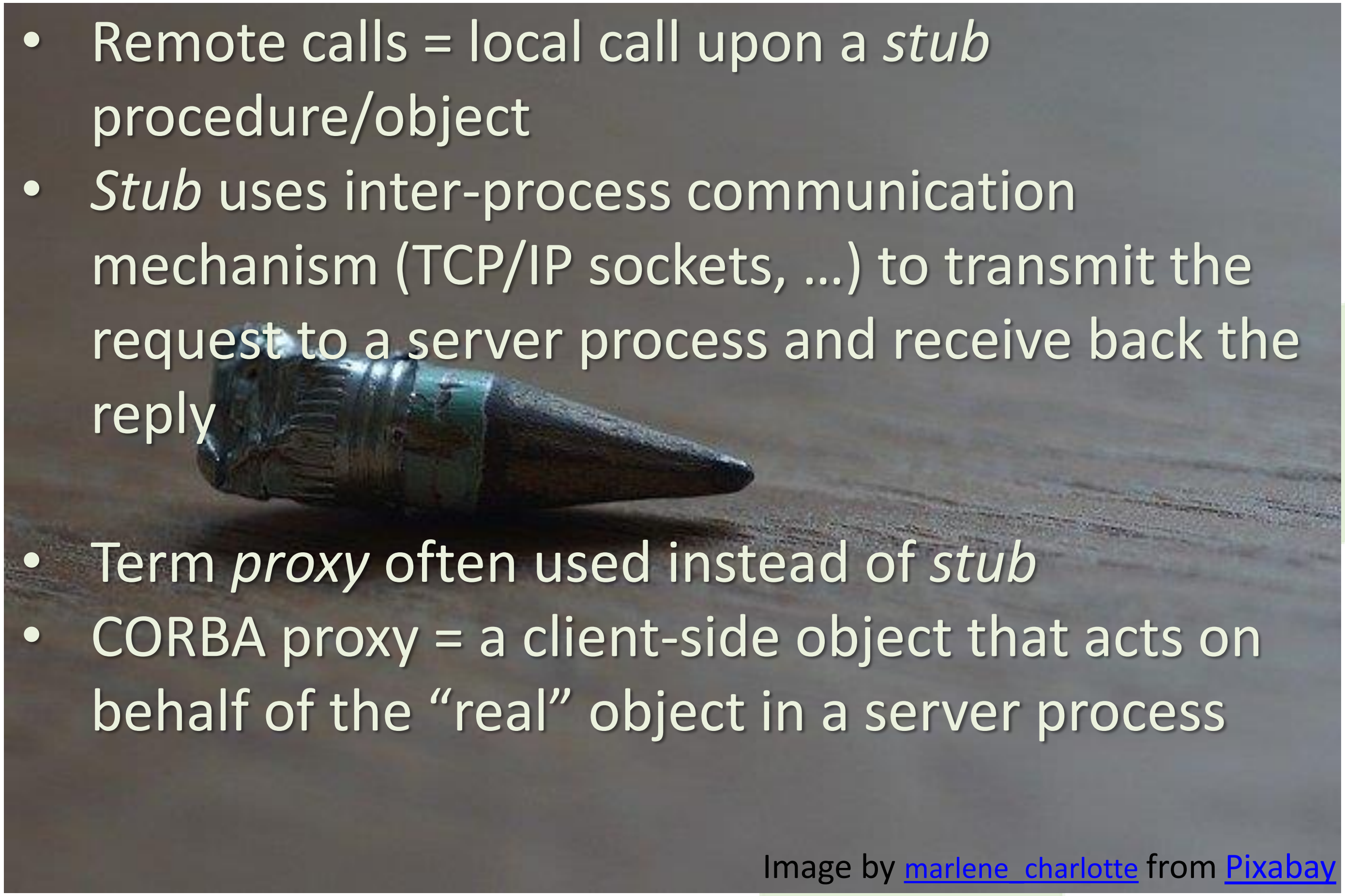
cppTango Dependencies: IDL

omniORB IDL Compiler = omniidl
CORBA standard defined mapping from IDL to programming languages



cppTango Dependencies:IDL

Stub/Proxy code



- Remote calls = local call upon a *stub* procedure/object
- *Stub* uses inter-process communication mechanism (TCP/IP sockets, …) to transmit the request to a server process and receive back the reply
- Term *proxy* often used instead of *stub*
- CORBA proxy = a client-side object that acts on behalf of the “real” object in a server process

Image by marlene_charlotte from Pixabay

Photo by Rae Tian on Unsplash

cppTango Dependencies:IDL

Skeleton code

http://gph.is/18HQbaL

- Skeleton code = server-side code for:
  - reading incoming requests
  - dispatching requests to application-level objects

cppTango Dependencies:IDL

Skeleton code

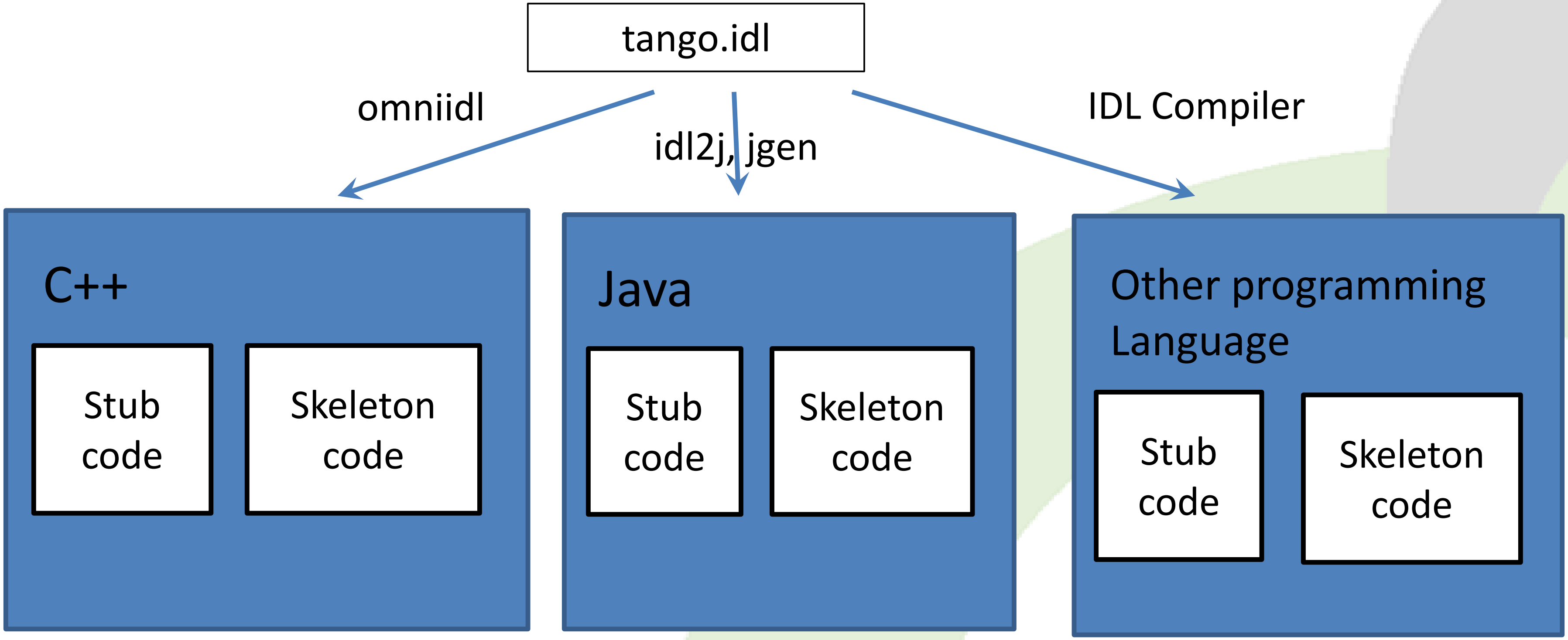
http://gph.is/2cZTNnv

- Skeleton code = server-side code for:
  - ➢ reading incoming requests
  - ➢ dispatching requests to application-level objects

- *Skeleton code* ⬄ **Supporting Infrastructure** required to implement server applications

cppTango Dependencies: IDL

omniORB IDL Compiler = omniidl (requires python)



cppTango Dependencies: CORBA

CORBA explained simply:
http://www.ciaranmchale.com/corba-explained-simply

How to compile cppTango?

Thomas Braun – ( ) byte physics

Photo by [Rohit Farmer]() on [Unsplash]()

Connecting things together

Thank you!

Any questions?

https://www.github.com/tango-controls/cppTango/issues

http://www.tango-controls.org/