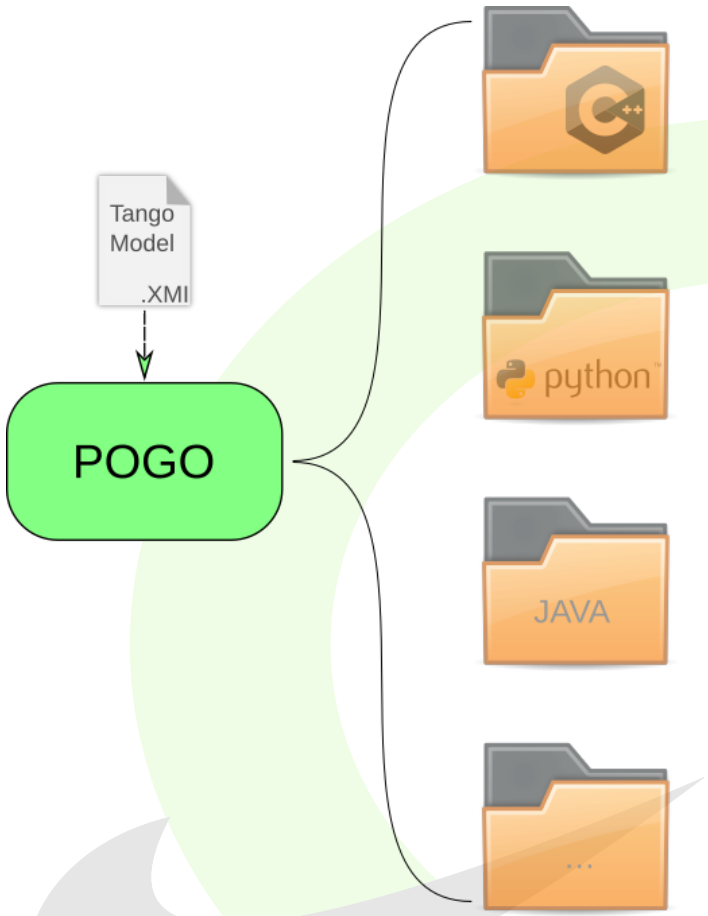
Pogo Roadmap implementation

Damien Lacoste

June 27 2023

POGO - Program Obviously used to Generate tango Objects



- Tango code generator written in java swing.
- Use a xmi file to describe tango class and device server model.
- Support device servers with several classes.
- GUI to create and manage tango class and device server model.
- Support generators for different languages and uses:
  - C++ code generator.
  - java code generator.
  - Python code generator, in 2 flavors.
  - HTML documentation generator.
- Run with java 11+.

Pogo roadmap

2 years ago... a proposal

What now?

Coming next...

Roadmap

- C++ generator code reorganization
  - Closer to Tango model.
  - Strong cmake integration.
  - Build libraries and binaries.
- CI/CD.
  - Clear release procedure.
  - Use of gitlab CI/CD pipelines to test.
- Get rid of "protected region" paradigm.
- Documentation.
- And maybe more...

A few ideas

- More powerful command line interface.
- Support other input formats.
- Rename fr.esrf.* java classes to org.tango.*.
- Move away from xtend/xtext technology.
- Pogo rewrite ?

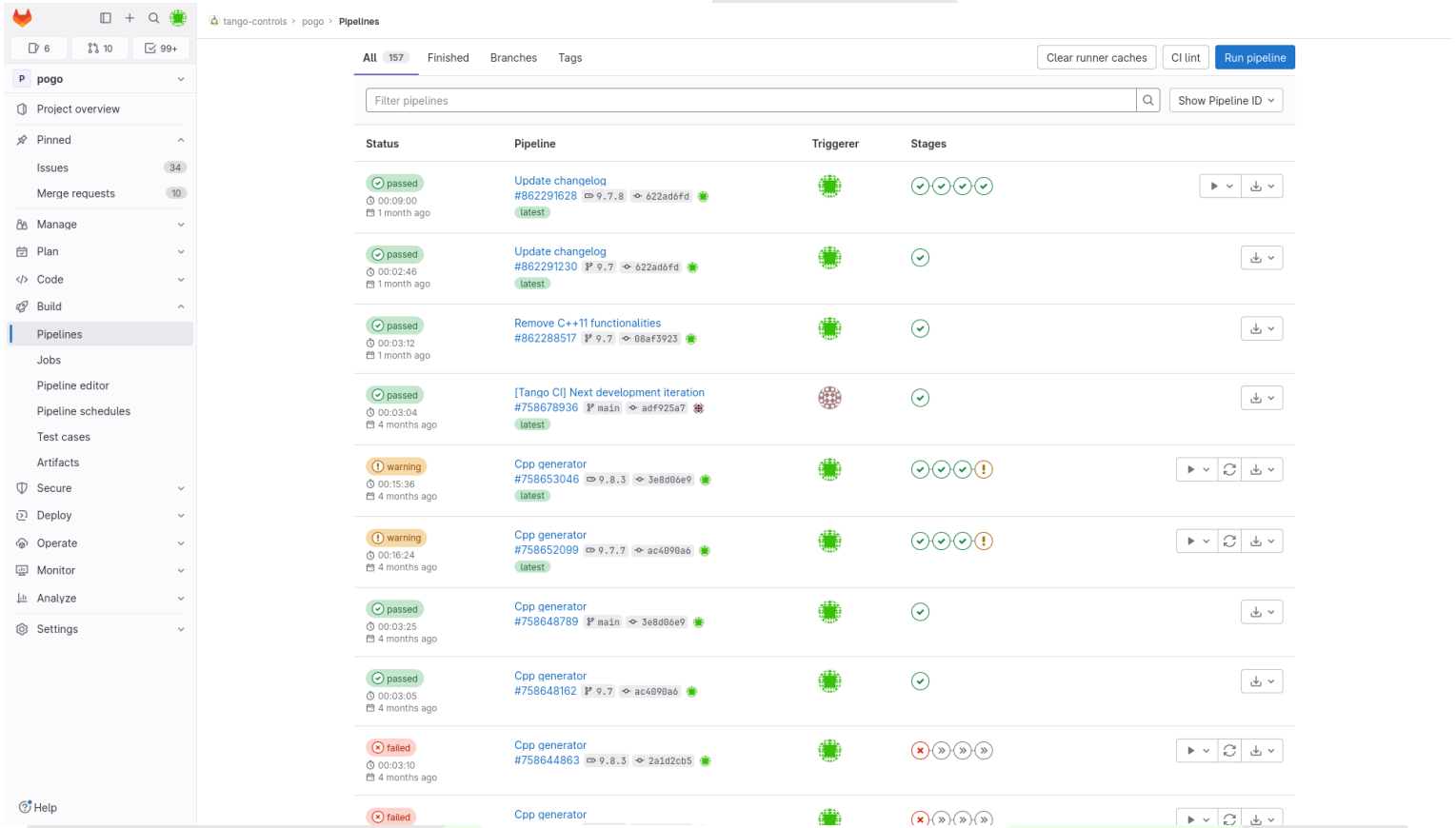
Pogo roadmap

2 years ago... a proposal

What now?

Coming next...

Pogo CI/CD


tango-controls › pogo › Pipelines
pogo
Project overview
Pinned
Issues 34
Merge requests 10
Manage
Plan
Code
Build
Pipelines
Jobs
Pipeline editor
Pipeline schedules
Test cases
Artifacts
Secure
Deploy
Operate
Monitor
Analyze
Settings
Help
All 157
Finished
Branches
Tags
Clear runner caches
CI lint
Run pipeline
Filter pipelines
Show Pipeline ID
Status
Pipeline
Triggerer
Stages
passed
Update changelog
#862291628 9.7.8 622ad6fd
latest
00:09:00
1 month ago
passed
Update changelog
#862291230 9.7 622ad6fd
latest
00:02:46
1 month ago
passed
Remove C++11 functionalities
#862288517 9.7 08af3923
00:03:12
1 month ago
passed
[Tango CI] Next development iteration
#758678936 main adf925a7
latest
00:03:04
4 months ago
warning
Cpp generator
#758653046 9.8.3 3e8d06e9
latest
00:15:36
4 months ago
warning
Cpp generator
#758652099 9.7.7 ac4090a6
latest
00:16:24
4 months ago
passed
Cpp generator
#758648789 main 3e8d06e9
00:03:25
4 months ago
passed
Cpp generator
#758648162 9.7 ac4090a6
00:03:05
4 months ago
failed
Cpp generator
#758644863 9.8.3 2a1d2cb5
00:03:10
4 months ago
failed
Cpp generator

C++ code generator refactoring: Pogo-9.9

```
± tree
.
├── classes
│   └── Test
│       ├── CMakeLists.txt
│       ├── include
│       │   ├── TestClass.h
│       │   └── Test.h
│       └── src
│           ├── TestClass.cpp
│           ├── Test.cpp
│           └── TestStateMachine.cpp
├── CMakeLists.txt
├── doc
├── servers
│   └── Test
│       ├── CMakeLists.txt
│       └── src
│           ├── ClassFactory.cpp
│           └── main.cpp
├── test
└── Test.xmi

9 directories, 11 files
```

- Clear separation between headers and source files.
- We can see the tango model structure of the classes and device servers.
- All projects are managed the same, whatever the number of classes they ship.
- Modern cmake integration with targets, and support for cmake installation.
- Full control about what to build, may it be a library, binaries, linking to static or shared library, activating support for part of the classes or not through the use of consistent cmake flags.

Pogo rewrite

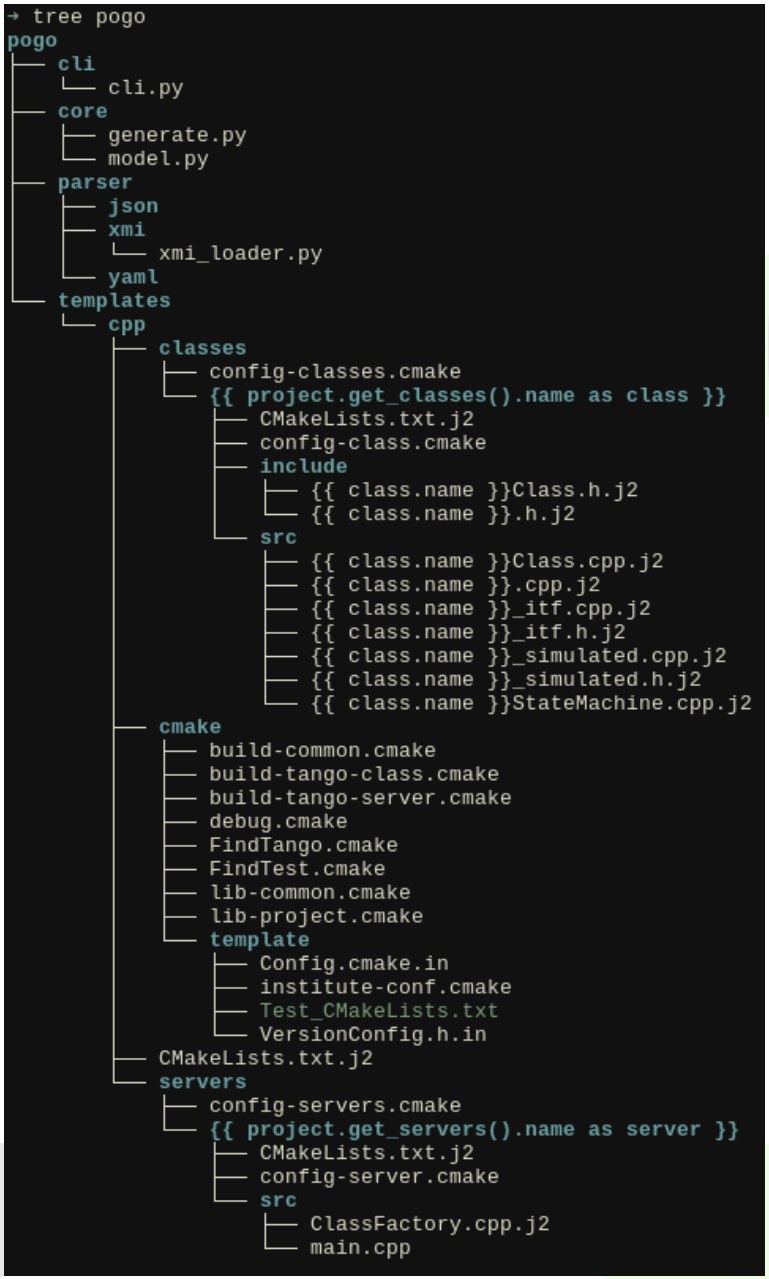
Python rewrite

- Use templates with jinja2.
- modular for the outputs and the inputs.
- support xmi as well as yaml and json.
- You never have to edit generated code.
- Generate a simulated device to test.

Old java implementation

- Rely on xtext/xtend, which is a niche framework.
- Monolithic hard to maintain app.
- Can only work with xmi.
- Rely on protected regions.

Pogo rewrite, a closer look



- Modular input, just load anything into a model. So far support for yaml, json and xmi.
- Input can actually be bidirectional (except for xmi).
- Modular templates for output too. Use the same input file to generate your code in whatever language.
- Write your own templates to extend pogo however you want.
- So far CLI only, but we could plug it to a GUI.
- Generate code for a simulated device.

Pogo rewrite, in case you're still reluctant

Python rewrite

- Execution time average for 100 runs.
  - 0.38s
- Lines of Code.
  - about 5000
- Github's stars for jinja2.
  - 9.3K

Old java implementation

- Execution time average for 100 runs.
  - 2.98s
- Lines of Code.
  - about 130000
- Github's stars for xtext.
  - 703
- Protected regions are a bad pattern!

Pogo rewrite, it's not all good

- Break compatibility, old devices will have to be rewritten, the code in the protected regions will not be ported.
- It's only a proof of concept, most of the work has yet to be done.
- And that's it...

Pogo roadmap

2 years ago... a proposal

What now?

Coming next...

A maintanable future

Python rewrite

- Write the templates for java and python.
- Implement other input formats.
- GUI or other tools integration.
- Support legacy (pre 11) C++.
- Get rid of Pogo alltogether!

Old java implementation

- Last java version, LTS.
- Java 17 migration.
- Could be used to help the migration.

Any Questions ?

Thanks !

Documentation

Project repository

https://isocpp.org

https://www.python.org

Jinja2 documentation