AGH

Status of work on the GGSS system

Tasks undertaken as part of the engineering and master's thesis

Arkadiusz Kasprzak
Jarosław Cierpich
Grzegorz Podsiadło

Supervisor: Bartosz Mindur

17th February 2020

Agenda

1. New project architecture and migration to GIT
2. New building system
3. Gitlab CI/CD
4. Documentation
5. Hardware tests
6. GGSS Reader and WinCC OA project changes for GGSS
7. Plans for future improvements

New project architecture

Characteristics of the new project architecture:

- Every module does only need minimal required dependencies to compile
- New architecture does bring valuable information about dependencies in the project and inter-module interactions
- Modules has been hierarchized. There are hierarchy levels and dependencies point only towards the lower level of hierarchy.

New project architecture

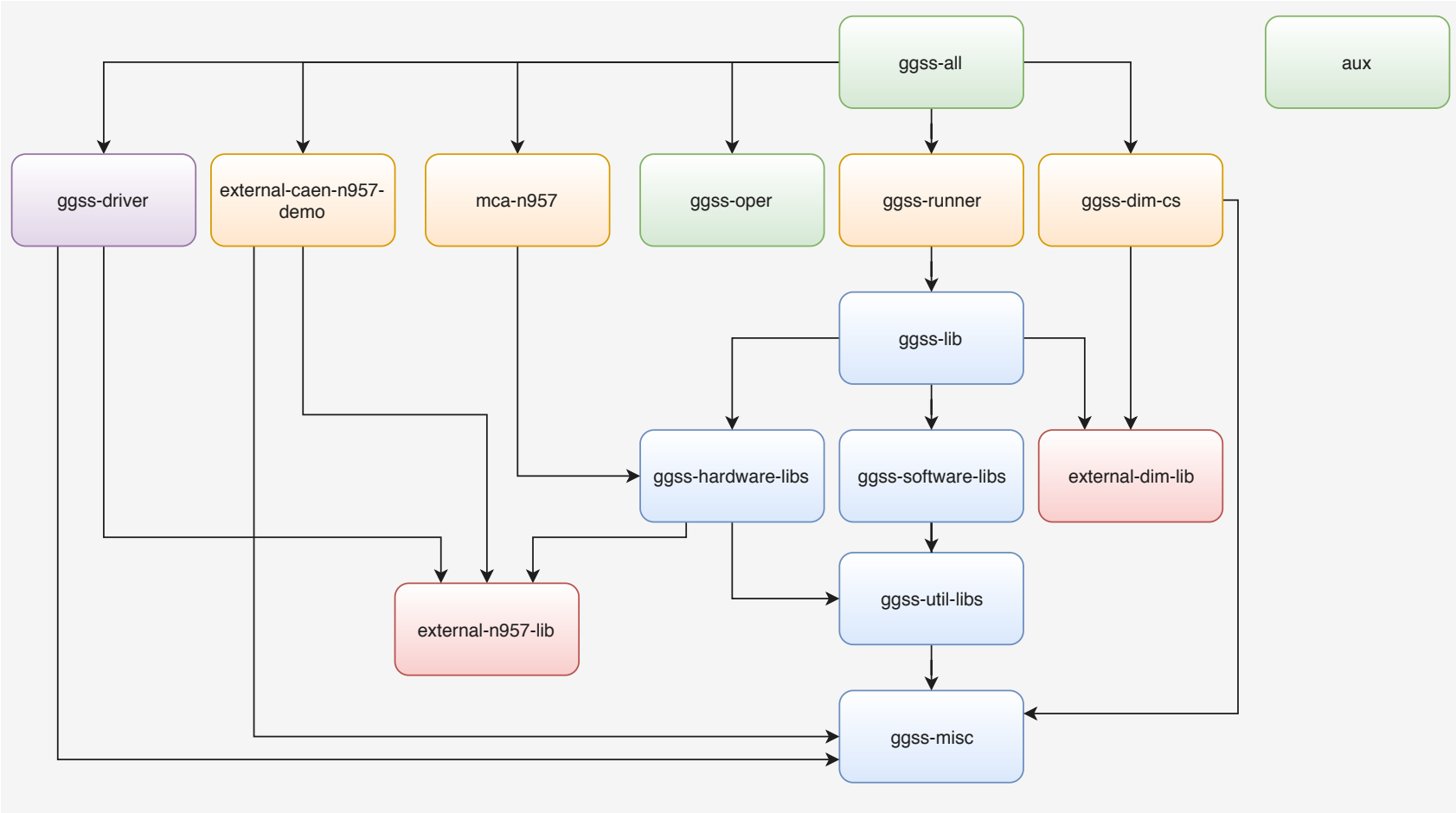


Figure: Architecture of the GGSS project

Migration to GIT

- Project has been migrated to GIT version control system. Every module has been divided into separate repository. Submodule feature has been used to achieve hierarchical structure and support fast setup of development environment.
- **atlas-trt-dcs-ggss** group has been created within which 20 repositories has been created.
- Issues, Milestones and Kanban Board are being used to organize and track work throughout development.

New building system

- New system based on CMake has been created.
- Hierarchical, information about dependencies clearly visible.
- Contains helper Python scripts - for example, in top repository, where user can choose which version should be built.
- System can easily be upgraded if some new requirements appear.

Gitlab CI/CD

- Continous Integration and Delivery environment has been created using Gitlab CI/CD.
- Building process of applications (ggssrunner, mca-n957 etc.) has been automated.
- Versions: static debug, static release, dynamic debug and dynamic release.
- Product can be downloaded using artifacts system.

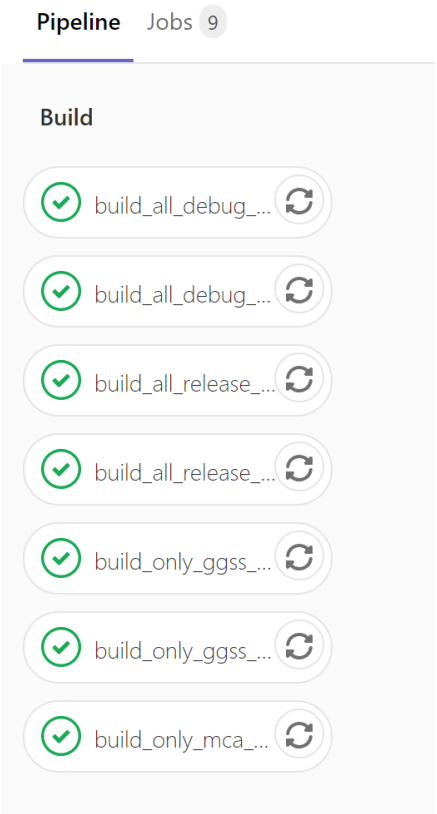


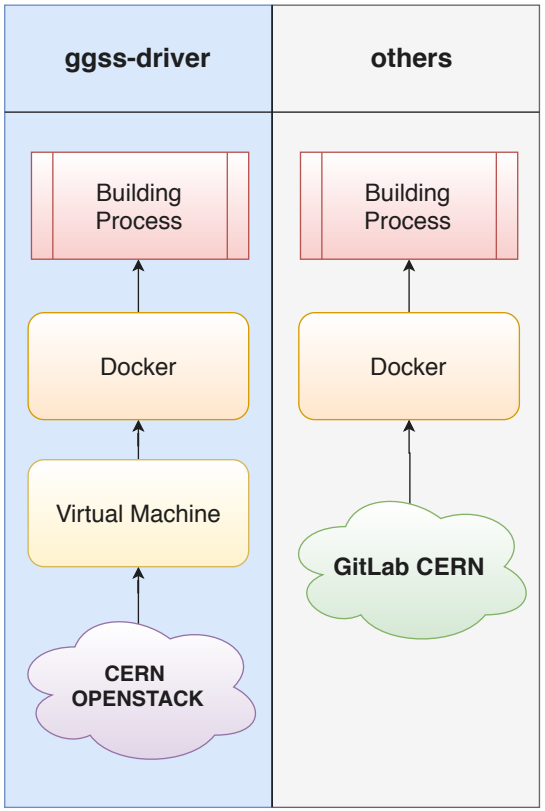
Figure: Pipeline used for the GGSS runner repository

Resources

Following resources has been used to establish building environment:

- GitLab CERN resources - to run CI/CD on every single repository except ggss-driver which requires control over installed kernel version
- OpenStack CERN resources - to run CI/CD for ggss-driver

Docker image has been prepared to achieve fast and reliable environment.



Documentation

- Documentation in english is being prepared.
- Readme files.
- Contains guidelines on how to build every component of the project.

Building whole project

To build all libraries at once, use the following commands:

- `git clone ssh://git@gitlab.cern.ch:7999/atlas-trt-dcs-ggss/ggss-software-libs.git` to clone the repository from GitLab
- `mkdir <build_directory>` where *build_directory* should be CMake output directory
- `cd ggss-software-libs`
- `git submodule update --init --recursive --remote`
- `cd ../<build_directory>`
- `cmake ../ggss-software-libs`
- `make`

Figure: Part of documentation that can be found in *ggss-software-libs* repository

GGSS and hardware tests

- GGSS project has been tested after changes to architecture and building system. Tests were successful
- Python scripts were prepared to test each hardware separately (High Voltage PSU, Multiplexer).
- These scripts, together with application dedicated to Multi Channel Analyzer test, were used to test hardware connection with evaluation machine (pcatltrteval) both via USB Hub and directly. Tests were successful

GGSS Reader

The GGSS Reader is a new tool that simplifies work on WinCC OA project development. Software allows to simulate the operation of GGSS using old measurements without the need for using hardware. GGSS Reader comes with the documentation and configured CI.

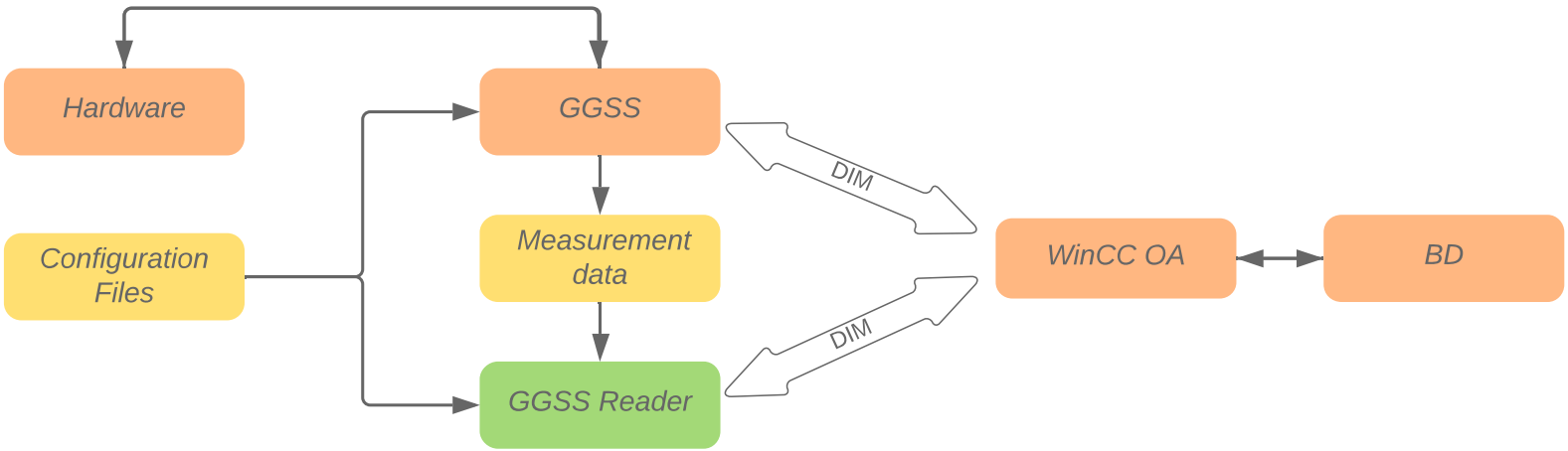
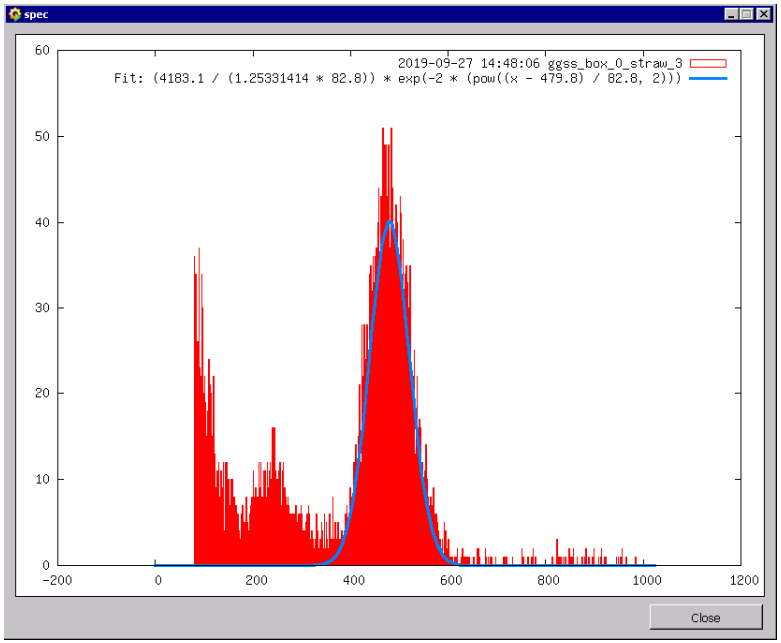


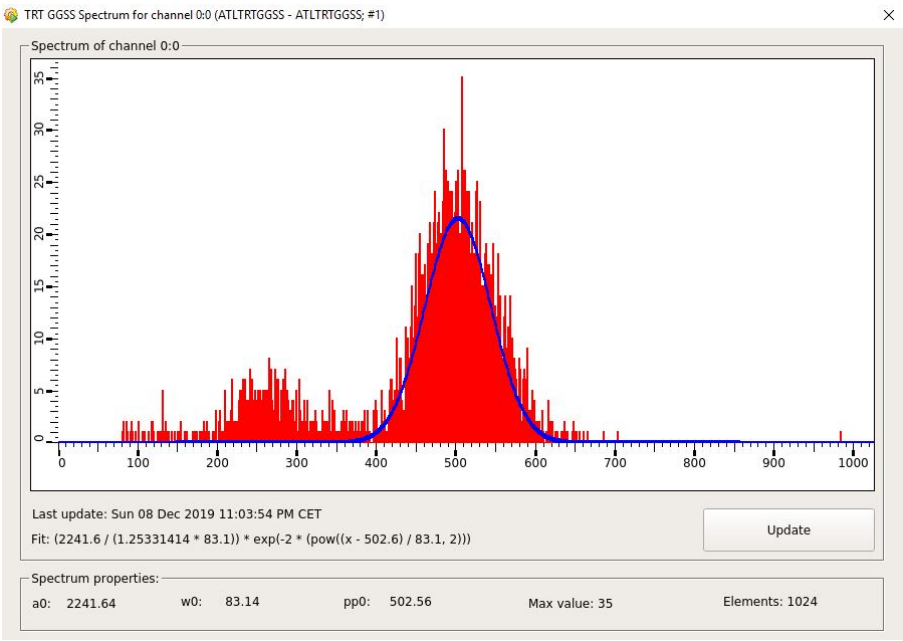
Figure: Infractructure related to GGSS.

WinCC OA project changes for GGSS

A new spectrum plotting panel was created, which allowed to get rid of external dependencies (gnuplot).



(a) Old version.



(b) Current version.

Figure: Comparison of spectrum plotting panels.

Plans for future improvements

- Automated versioning (master branches version align)
- Possibility to reuse Continuous Delivery mechanism with debug builds (possibility to use different branch than master to build project)
- Code refactoring (for example, include paths).
- Extended GGSS responsiveness - system parameters updated on demand and at the beginning of execution.
- Improvements in curve fitting algorithm.
- More options for the GGSS Reader program and a graphical interface for easier data selection.

Thank You! Questions?