



AGH UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Update and upgrade of the GGSS system for ATLAS TRT detector

Arkadiusz Kasprzak
Jarosław Cierpich

Supervisor: Bartosz Mindur



- 1 A brief introduction to GGSS**
- 2 Tasks and constraints**
- 3 Overview of changes**
- 4 Plans for future improvements**



AGH Introduction to GGSS

- Gas Gain Stabilization System (GGSS) is a project integrated with ATLAS Detector Control System (DCS).
- It helps to ensure proper operation of the TRT (Transition Radiation Tracker) detector, which is a part of the ATLAS detector at CERN.
- GGSS consists of hardware and software layers.
- Today we will focus mainly on the software layer.
- Most of the codebase is written using C++. There is also some Python and Bash code.

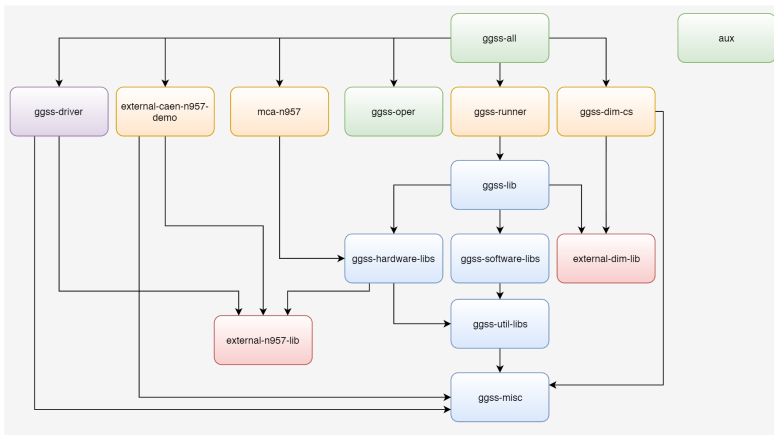


Figure: Software components of the GGSS project and their dependencies.



- C++ codebase refactoring:
 - migration to C++11/14 (range-for loops, uniform initialization etc)
 - removing old, unused code
 - adding more comprehensive documentation
 - introducing TDD (Test Driven Development)
- CMake files refactoring
- creating tools for versioning and Git submodule handling



- The user must be able to build the project using the CERN infrastructure
- Code refactoring had to be performed with this constraints in mind.
- Only old version of CMake (2.8) and GCC (C++11/14, but no 17/20) available.



Example of migration to C++11/14 - replacing iterator loop with range-for one.
Below You can see the old code.

Listing 1: Example of old C++ code (before refactoring).

```
for ( XMLTag::NestedTags::const_iterator j = tag.getNestedTags().begin()
      ; j != tag.getNestedTags().end()
      ; j++
    )
{
    if((j->second->getName() == tagName)
        &&(j->second->getAttributeValue("id") == idValue))
        return j->second;
    else
        m_findTagById(*(j->second), tagName, idValue);
} // endfor
```



Listing 2: Example of new C++ code (after refactoring).

```
const XMLTag::NestedTags& nestedTags = startingTag.getNestedTags();
for(const auto& nestedTag: nestedTags)
{
    if((nestedTag.second->getName() == tagName) &&
        (nestedTag.second->getAttributeValue("id") == idValue))
    {
        return nestedTag.second;
    }
}
```

- using range-for loop increases readability of the code
- else clause has been removed - result of the recursive function call was never used
- no need to use the * operator
- nestedTag is a better name than j



AGH Removing old, unused code

- The project contained a lot of code (functions/methods) that were never used.
- Some of them could even be harmful if used.
- Below example shows two methods that have been removed (why?) from QueueLimited class (a queue with size limit).

Listing 3: Example of removed code.

```
// return the whole queue
const std::deque<T>& getQueue () const {
    return c;
}

// return the whole queue
std::deque<T>& getQueue () {
    return c;
}
```



AGH Introducing Test Driven Development

- For unit tests, we are using Boost.Test, because it is simple and available when using CERN infrastructure.
- Component are tested during refactoring, we make sure that our changes do not introduce any new bugs.
- Each component can be tested separately.

Listing 4: Unit test example

```
/**
 * \brief Checks if proper exception is thrown when performing pop()
 *         operation on empty container.
 */
BOOST_AUTO_TEST_CASE(
    testIfExceptionIsThrownWhenTryingToPopFromEmptyContainer)
{
    QueueLimited<int> queue{};
    BOOST_CHECK_THROW(
        queue.pop(),
        QueueLimited<int>::ReadEmptyQueueException);
}
```



AGH Continuous Integration

- Practice widely used during modern software development.
- Developers integrate code into repository frequently.
- Each code contribution is automatically built and tested.
- This allows for quick error detection.
- We use GitLab CI/CD for building and testing the project.

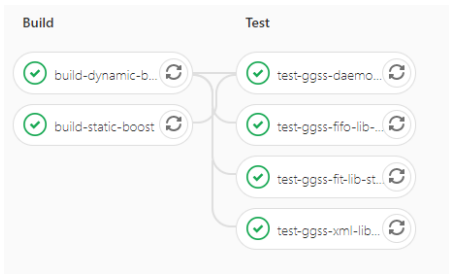


Figure: Example of CI pipeline used in the project.



AGH CMake files refactoring

- GGSS uses CMake for managing the build process of the software.
- CMake is platform and compiler independent.
- CMake files have been slightly refactored to improve readability by using macros and functions.

Listing 5: Old version of CMake used for building *thread-lib*

```
set(target_name "thread")
if(NOT TARGET ${target_name})
  set(CMAKE_MODULE_PATH "${GGSS_MISC_PATH}")
  include(BuildLibrary)
  include(FindLibraryBoost)
  include(SetupDoxygen)
  include(SetupTests)

  # notice the need to set some variables before including the file
  set(dependency_prefix "${CMAKE_CURRENT_SOURCE_DIR}/..")
  set(dependencies "handle" "log")
  include(BuildDependencies)
endif()
```



AGH CMake files refactoring

- Instead of including the CMake template files (which just pastes the code), we invoke `ggss_build_library` with named parameters.
- Unit tests and Doxygen support has been moved to `ggss_build_library` macro, because every library in the project uses them.

Listing 6: New version of CMake used for building *thread-lib*

```
set(CMAKE_MODULE_PATH "${GGSS_MISC_PATH}")
include(BuildLibrary)

ggss_build_library(
  TARGET_NAME "thread"
  DEPENDENCY_PREFIX "${CMAKE_CURRENT_SOURCE_DIR}/.."
  DEPENDENCIES "log" "sigslot"
)
```



AGH Complex submodule structure handling - scripts

- GGSS project tree contains a complex repository structure with many connections between components.
- To make it easy to properly initialize project structure git submodules are being used.

Listing 7: Initialize project structure with one command.

```
root@host:/# git clone
  ssh://git@gitlab.cern.ch:7999/atlas-trt-dcs-ggss/ggss-all.git && cd
  ggss-all && git submodule update --init --recursive
Cloning into '/CERN/ggss-all/ggss-dim-cs' ...
Cloning into '/CERN/ggss-all/ggss-driver' ...
Cloning into '/CERN/ggss-all/ggss-oper' ...
Cloning into '/CERN/ggss-all/ggss-runner' ...
Cloning into '/CERN/ggss-all/ggss-spector' ...
Cloning into '/CERN/ggss-all/mca-n957' ...
Cloning into '/CERN/ggss-all/ggss-dim-cs/external-dim-lib' ...
Cloning into '/CERN/ggss-all/ggss-dim-cs/ggss-misc' ...
Cloning into '/CERN/ggss-all/ggss-driver/external-n957-lib' ...
Cloning into '/CERN/ggss-all/ggss-driver/ggss-misc' ...
...(13 lines truncated)
```



AGH Complex submodule structure handling - scripts

- Using submodules requires to take care of commit hashes that are being linked as a submodule.
- There may be a situation that "parent" repository is not using the latest version of "child" repository.

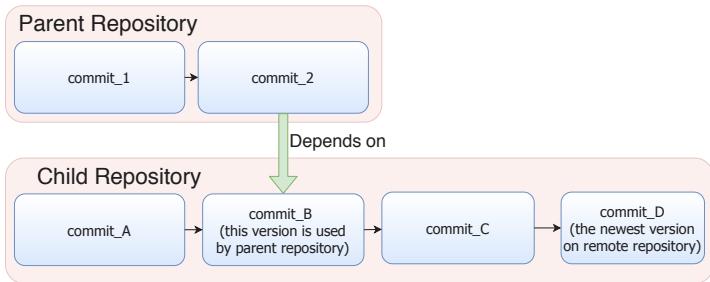


Figure: Version of submodule differs from version used by parent.



AGH Complex submodule structure handling - scripts

- gitio script is responsible for updating all outdated links between parent and child repositories.
- The goal is achieved by creating dependency tree of all available repositories.
- Starting from the bottom of the tree submodules are being aligned (git commands: add, commit, push).

Listing 8: Gitio in action.

```
root@host:/# python gitio.py -p ./ggss-all/  
...(17 lines truncated)  
INFO - Aligning ./ggss-all/mca-n957 repository  
INFO - Aligning ./ggss-all/ggss-dim-cs repository  
INFO - Aligning ./ggss-all/ggss-runner repository  
INFO - Aligning ./ggss-all/ggss-spector repository  
INFO - Aligning ./ggss-all/ggss-oper repository  
INFO - Aligning ./ggss-all/ggss-driver repository  
INFO - Aligning ./ggss-all repository  
INFO - Aligning finished.
```




AGH Automated versioning

- Automated versioning system has been prepared to keep consistent rpm and release versions throughout whole project.
- Every commit to main repository (ggs-all) is being analyzed. If commit message contains one of specified phrases, new release is being created.

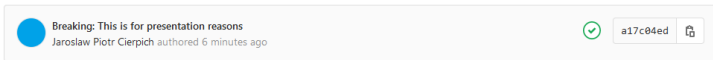


Figure: New commit following eslint convention.

```
[2:49:15 PM] [semantic-release] [@semantic-release/commit-analyzer] > i Analyzing commit: Breaking: This is for presentation reasons
[2:49:15 PM] [semantic-release] [@semantic-release/commit-analyzer] > i The release type for the commit is major
[2:49:15 PM] [semantic-release] [@semantic-release/commit-analyzer] > i Analysis of 29 commits complete: major release
[2:49:15 PM] [semantic-release] > ✓ Completed step "analyzeCommits" of plugin "@semantic-release/commit-analyzer"
[2:49:15 PM] [semantic-release] > i The next release version is 1.0.0
```

Figure: Commit message analysis.

v1.0.0

▼ Assets 4

-  Source code (zip)
-  Source code (tar.gz)
-  Source code (tar.bz2)
-  Source code (tar)

Evidence collection

-  [v1.0.0-evidences-1616.json](#) ... 2d0c3ef6 
-  Collected 3 minutes ago

1.0.0 (2020-09-21)




 [3a8fb430](#)  [v1.0.0](#) Created 3 minutes ago by 

Figure: Newly created release.



AGH Plans for future improvements

- Further code refactoring.
- Introducing new features, for example on-start and on-demand GGSS parameters update.
- Creating RPM package with whole project - easy deployment.
- Improving Python library and apps for hardware testing. E.g. yaml scenarios.



Thanks for Your attention.