AGH University of Science and Technology

Core GGSS software update and upgrade
Tasks undertaken as part of the master's thesis

Arkadiusz Kasprzak
Jarosław Cierpich

Supervisor: Bartosz Mindur

Overview of changes

- C++ codebase refactoring:
  - migration to C++11/14 (range-for loops, uniform initialization etc.)
  - conventions unification (code formatting, function naming etc.)
  - removing old, unused code
  - introducing TDD (Test Driven Development)
  - adding more comprehensive documentation
- project structure refactoring (removing unused dependencies, renaming files and one library)
- CMake files refactoring
- creating tools for Git submodule handling
- creating tools for versioning

C++ codebase refactoring

- 12 out of 14 main libraries in the project received some kind of code refactoring
- unified code and documentation convention has been applied in every library
- some Boost features have been replaced with their C++11 counterparts
- deprecated and not recommended parts of code have been modernized (using range-for loops, noexcept, uniform initialization etc.)

Listing 1: 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;
    }
}
```

C++ codebase refactoring

- the project contained a lot of code (functions or even whole classes) that were never used
- unreachable and commented out code has been removed
- unused parts of code (for example else branches or unsafe methods) have been removed
- below example shows two methods that have been removed from `QueueLimited` class (a queue with size limit).

Listing 2: 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;
}
```

Introducing Test Driven Development

- For unit tests, we are using `Boost.Test`
- Components are tested during refactoring, we make sure that our changes do not introduce any new bugs.
- Each component can be tested separately.

Listing 3: 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);
}
```

Continous Integration and TDD

- Unit tests have been integrated into out CI/CD infrastructure.

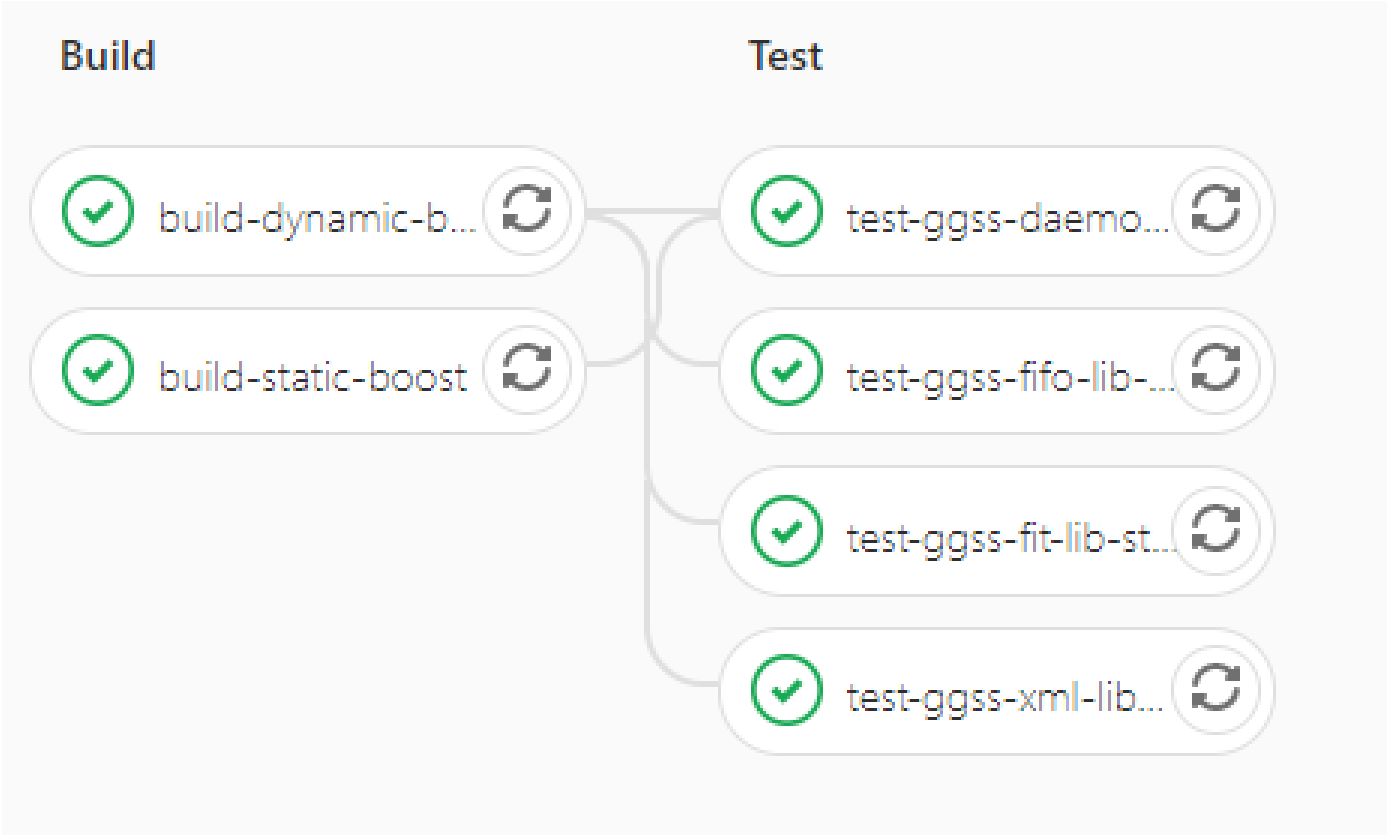

Figure: Example of CI pipeline used in the project.

CMake files refactoring

- CMake files have been slightly refactored to improve readability by using macros and functions.
- Doxygen and unit testing support have been added.

Listing 4: 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"
)
```

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 5: 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)
```

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.

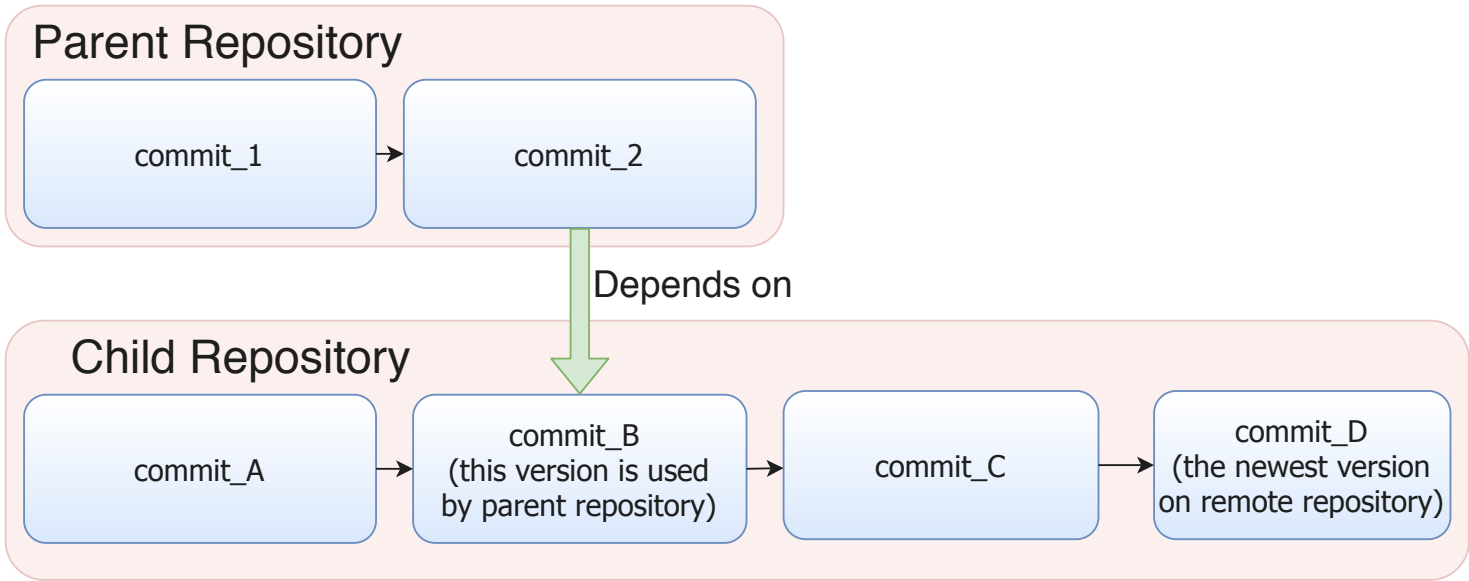

Figure: Version of submodule differs from version used by parent.

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 6: 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.
```

Automated versioning

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

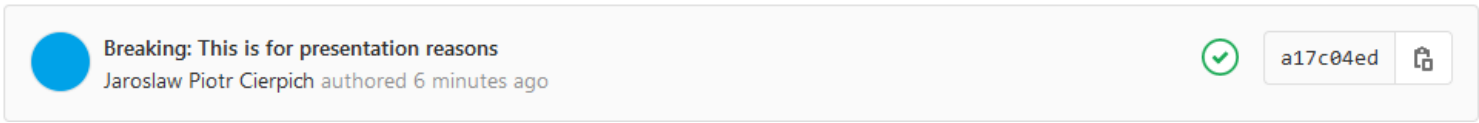

Figure: New commit following eslint convention.

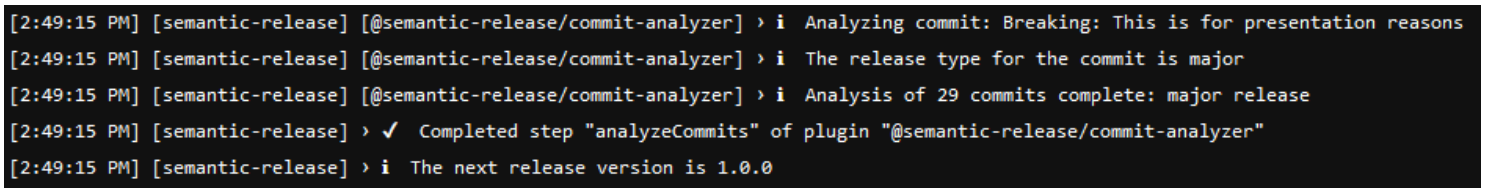

Figure: Commit message analysis.

Automated versioning

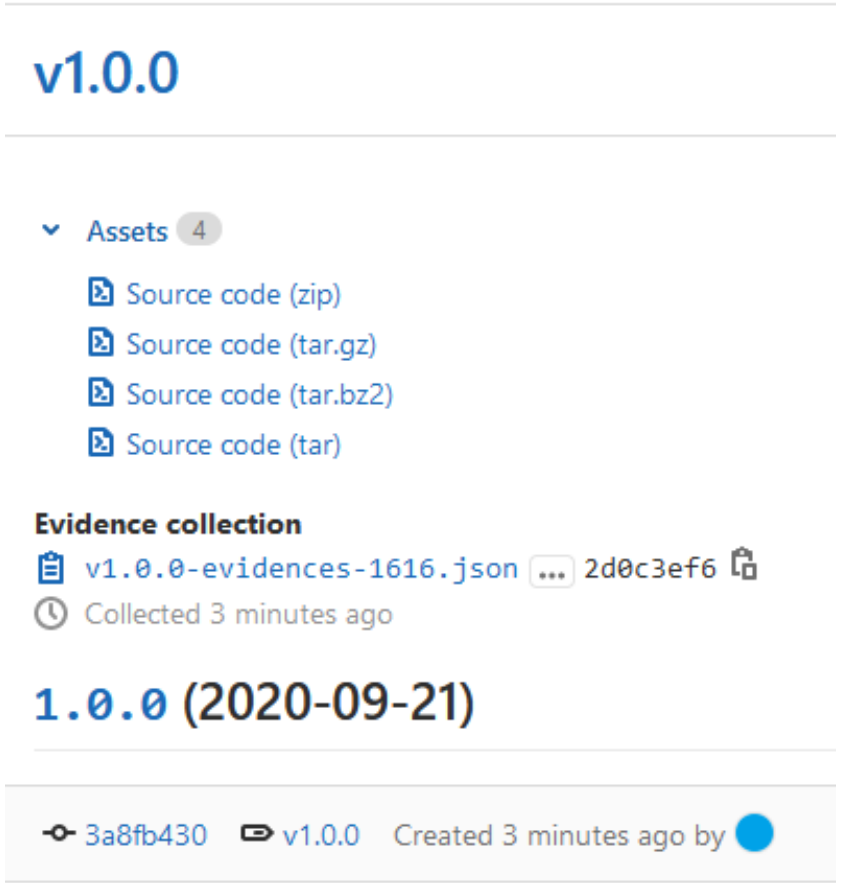

Figure: Newly created release.

Work in progress

- Hardware testing scripts refactoring and upgrade (yaml based scenarios).
- Gitio script improvements (interactive entry point, automated repository structure initialization).
- Custom artifacts (RPM package, compiled code, documentation) attached to releases.
- HV management commands refactoring (user-friendly format for SET and MON commands).

Thanks for Your attention.