The current ACS Makefile has been improved over the years allowing a per-module parallel build, however it has always been an issue to build the whole system efficiently. There have been improvements over the years (like building subsystems in parallel), but we're far from a lean process when small changes are made to some module. It actually means to build everything again, leading to an ~8 hours compilation process.

There are at least 3 main areas where we aim to improve:

- Dependencies Calculation: This helps in the creation of patches, without missing updating libraries or dependent code, because we didn't force to build some dependencies we were not aware of. It could also be used at some point to create diagrams automatically, which may help to estimate the impact of a new change in the code. It is also a requirement for continuous integration.
- Inclusive Makefile: Currently, we make use of a recursive strategy to build all the subsystems, but the problem is that this isolates the dependencies calculations to single modules. The idea is to make use of an inclusive makefile, which goes through the subsystems trees, parsing the makefiles, leading to a single process with the information from all modules.
- Continuous integration: This would allow to considerably lower the building times, as only the change and its dependencies would need to be recompiled. This is very useful to maintain a build always compiling, and one could also trigger testing jobs related with the changes.

Pros and Cons

- Based on GNU Makefile, similar to the existing Makefile
    - Experience and knowledge of the syntaxis and language
    - Provides all the capabilities to achieve the desired objectives
    - Can coexist with the original Makefile
- Heavily based on Makefile definitions
    - Reduces code duplication
    - Small drawback is in argument identification ($1, $2, $3 ...)
        - Could be mitigated by assigning names: $(eval variable1=$1)
        - Meaningful names could be configured in this way
        - Something like: $(call setVariables,name dir ext) could map to something like $(foreach var,$1,$(eval $(var)=

Description