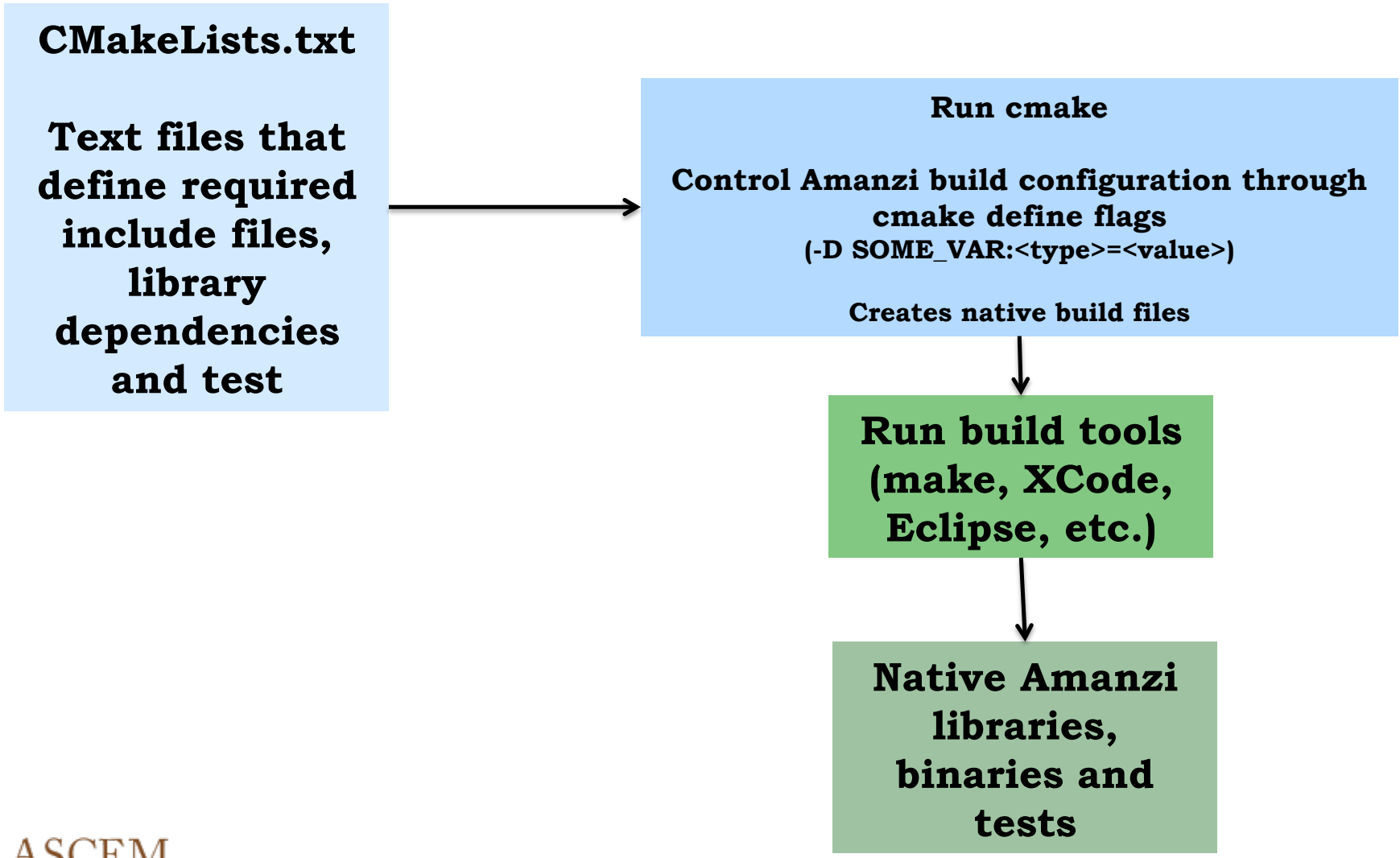
Amanzi Build System

- Two stages to build Amanzi.
- Build required and optional Third Party Libraries (TPLs)
  - Will have tools to build and configure TPLs (BuildSystem)
  - Currently, most developers use shell scripts to consistently build and configure. Some samples available in tools/config.
  - Will have a document detailing all required and optional libraries. A temporary list that documents the TPLs can be found in the Amanzi document directory doc/core/amanzi_tpl.tex.
- Build the Amanzi libraries and drivers.
  - Requires CMake 2.8 or higher.
  - BuildSystem will create an include file that will define the required variables that the Amanzi CMake files need to locate the TPLs.
  - Currently , most developers have a do-amanzi-configure shell script that contains all the '-D' options needed to build Amanzi. Examples of these shell script can be found in tool/config.

CMake Work Flow



ASCEM

ascemdoe.org

CMake Primer

- CMake is an open source tool that creates native build files for several platform types (UNIX, Linux, Mac OSX, Windows)
  - It is the cross platform support that Amanzi needs!
  - Part of a suite of tools (CTest, CDash, CPack) available from Kitware
- CMake is a macro scripting language.
- Control the build through text configuration files (CMakeLists.txt, FindXXX.cmake, SomeFile.cmake etc.)
- Generate the user manual in HTML format by executing:

  cmake --help-html=foo.html

  - Complete documentation of commands, variables and modules available will be listed in this file.
- CMake mail list is an active and responsive community. Will answer advanced and noob questions.
  - Subscribe at http://www.cmake.org/mailman/listinfo/cmake
  - Search email list through Google.

CMake Primer

- An interactive tool: ccmake (See cmake)
  - Available when cmake is built with gui support
  - Same command line options as cmake
  - Allows the user to view all the variables in the configuration and build and adjust those values before configuration.
- Generate a dependency graph for the graphviz tool with –graphviz=[file] option.
- To view the explicit compile command set VERBOSE=1,on,True when launching the build. (Example: make VERBOSE=on)
- The 'help' target will show all possible targets of the build system.
- The CMake manual is generated with the –help-html option.
- CMake lists are strings with entries separated by semicolons ';'

CMake Primer: CMake != GNU Auto*

- CMake is focused on one task: Generate native build files.
- GNU build system is primarily two tools, autoconf (configure script generation) and automake (Makefile generation).
- GNU is *very* UNIX-centric. No native Windows support.
  - Linking relies on LD_LIBRARY_PATH, other environment variables and explicit "-L/some/path –lbar" strings.
- CMake builds dependency chains (i.e. foo needs libA.a and libB.a: target_link_libraries(foo A), target_link_libraries(A B) builds the chain)
- GNU Auto* has a global namespace, Cmake is scoped.
- CMake will allow circular dependencies! (target_link_libraries(A B) and target_link_libraries(B A)
- CMake functions and macros are not case sensitive.
- CMake has three true expressions: 1, ON, True

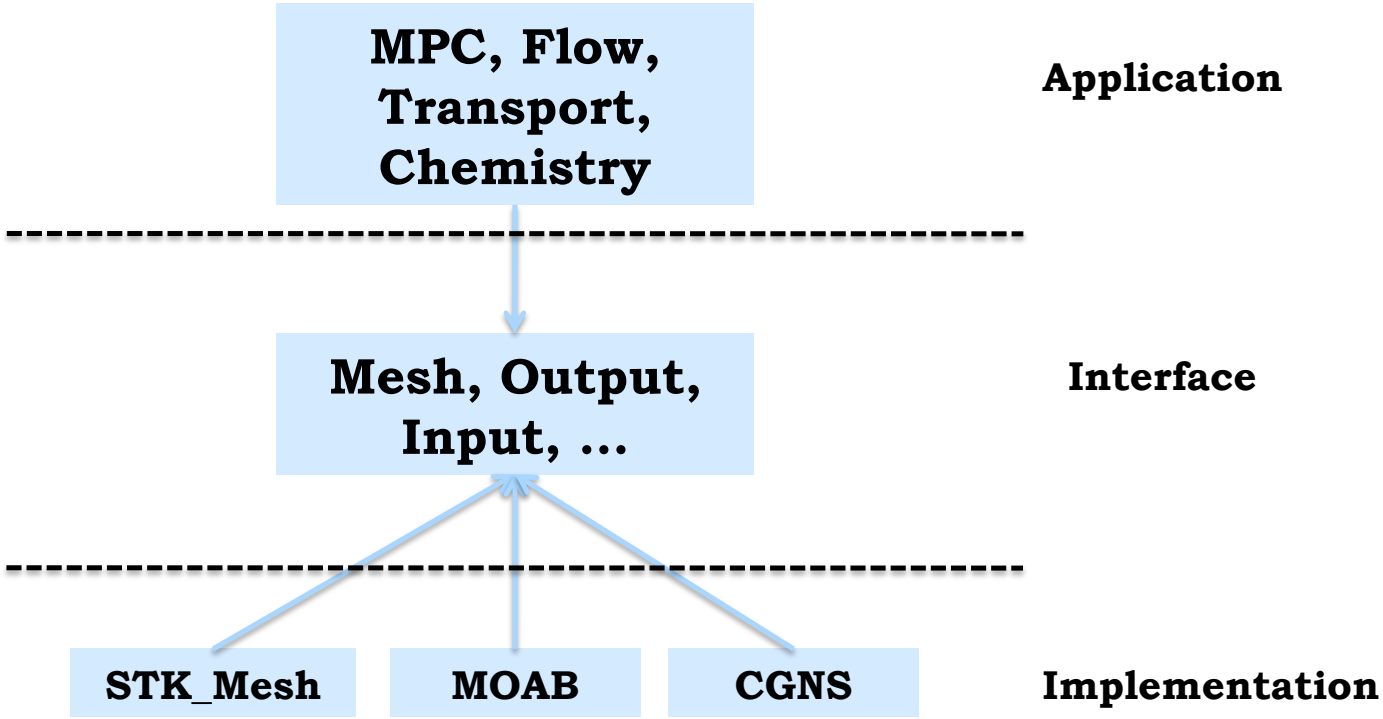
Amanzi Build System: Defining TPL Tragets and Libraries

- All TPLs have either a FindXXX.cmake or a XXXConfig.cmake file that defines the include paths and libraries needed to compile and link against XXX.
- FindXXX.cmake are CMake modules for software libraries that are not built with CMake and must define the following variables
  - XXX_INCLUDE_DIR Path to XXX include directory
  - XXX_LIBRARY_DIR Path to XXX library or libraries
  - XXX_INCLUDE_DIRS A list of all required include paths to compile XXX
  - XXX_LIBRARIES A list of targets or libraries needed to link against XXX
- Libraries built with CMake generate XXX-config.cmake or XXXConfg.cmake files and define these variables without calling find_file or find_library.
- The command find_package(XXX) searches for both kinds of files, FindXXX.cmake first then configuration files.
- When the command returns, XXX_FOUND is a boolean flag that indicates if the find was successful and if the variables above are set.
- Use the plural variables (XXX_INCLUDE_DIRS and XXX_LIBRARIES) to define include paths and link targets.

Amanzi Build System: CMakeLists.txt

- CMakeLists.txt are text configuration files that cmake needs to create build files.
- Each directory is defined as a project with the command project(FOO)
  - This defines useful variables: FOO_SOURCE_DIR (source file location) and FOO_BINARY_DIR (build directory location) visible to other project directories.
  - These variables are NOT defined until after add_subdirectory(foo) returns. This requires that subdirectories are called in the correct build order.
- Each subdirectory has a test directory. For examples, create an examples directory and place the source in that directory.
- Sample CMakeList.txt located in the tools/cmake directory. Follow this template when creating a new CMakeList.txt file.
- If you create a library you must include at least one test to demonstrate that an executable can link against the library. This is the best way to ensure that the dependencies for this new library are defined correctly.

How To Define Dependencies



If your target object requires crossing more than one layer to build then there is either a missing dependency definition or code needs to define an interface.

How To Define Dependencies

- Example: Need to build library foo that uses the Amanzi error handling library, Amanzi mesh class and CGNS output class.
- Developer knows that the Amanzi objects depend on other parts of Amanzi and TPLs. If the build system is functioning correctly, then the developer does NOT need to specify these other dependencies.
- Let CMake do its job.
- Bug reporting and fixing improves the design.

```
# Use file glob to reduce editing needs.
# Developers can add/change/remove source files
# and are not required to alter CMakeLists.txt if the
# file name matches this regular expression.
file(GLOB foo_source_files foo*.cc)

# This creates a foo target.
add_library(foo ${foo_source_files})

# This defines foo dependencies.
# Use variables defined in other CMakeLists.txt files
# to reduce the need for changes when libraries are
# moved or renamed.
target_link_libraries(foo ${DBC_LIBRARY}
                          ${AMANZI_MESH_BASE_LIBRARY}
                          ${AMANZI_OUTPUT_LIBRARY} )

# Tests are an excellent way to flush out dependency bugs.
# Notice that the target link command does not contain ANY
# other dependencies but library foo and UnitTest.
# If this executable does not build then there is an error in
# the build system.
If (ENABLE_TESTS)
    add_executable(test_foo test/Main.cc test/test_foo.c)
    target_link_libraries(test_foo foo ${UnitTest_LIBRARIES})
    :
```

Amanzi Build System: How To Make Changes

- Find a place for your code before altering the build system.
- Directories at the top of the source tree are reserved for components needed for the Multi-Process Coordinator (MPC) OR the functionality is required by two or more modules found in the top directory.
- Co-locate source files with the source using the class/data/whatever your code is creating.
  - Stand-alone binaries (amanzi-driver, mesh auditors) will be built in a top-level bin directory. The source for these binaries is located in src/executables.
- Maintain the current CMakeLists.txt format.
  - Use file globing whenever possible.
  - Global properties are in ALL CAPS.
  - Use the TPL variables and global variables defined in other CMakeLists.txt files.
  - Do not use explicit paths.
  - Reference the sample found in tools/cmake/SampleCMakeLists.txt and other working CMakeLists.txt files

Amanzi Build System: How to Make Changes

- If your object (library or executable) will be referenced outside of your current source directory, create a global property so that other CMakeLists.txt files can access the name of the object through a variable.
  - Remember CMake is scoped! See the Sample CMakeLists.txt file or other CMakeLIsts.txt files in the src directory.
- If you need to make significant changes to the build system, please test those changes on the build_system_changes branch before merging you changes into the default repository.
  - We will ask that developers test significant build branches changes on at least two systems before merging into the default.
  - Significant changes: Refactoring or adding new Amanzi*.cmake files, new TPL FindXXX.cmake modules or removing/adding new directories in the source tree.

Ask questions before you hack!

- If you can not build, there is a bug and it needs to be fixed. Bug reporting and fixing improves the design.
- HPC Core Team contacts
  - Lori A. Pritchett (lpritch@lanl.gov)
  - Richard Mills (rmills@ornl.gov)
  - Michael Buksas (mwbuksas@lanl.gov)
- Check the CMake manual (cmake –help-html)
- Search the CMake mailing list. (Google)
- Post a question to the Amanzi developers list (ascem-hpc-dev@lanl.gov)
- Post a question to the CMake email list. http://www.cmake.org/mailman/listinfo/cmake