Enabling assembly level structural coverage

The project under test is FreeCiv, a popular open source 4X strategy game. We have forked the original Github to include Parasoft C/C++Test specific sources and properties. Find it here: https://github.com/parasoft/freeciv/tree/MCDC_UT_example. The FreeCiv project was selected due to the use of conditional constructions analogous to the statistics reported by "An Investigation of Three Forms of the Modified Condition Decision Coverage (MCDC) Criterion" appendix C.

Initial expectations:

This guide assumes the Freeciv project is setup and buildable before generating or executing unit tests. Please verify that the original project will generate executables for linux x86_64 using GNU GCC.

FreeCiv Unit Testing filesystem listing:

- Properties files:
  - .cproject is configured to support building, running, and debugging the project.
  - .parasoft is configured by default to use GNU GCC 9.x (x86_64) modification will be necessary if another version is being used.
- Source files under test:
  - ${project_loc}/common/metaknowledge.c
- Unit test harness source files:
  - ${project_loc}/common/tests/metaknowledge_tests.c
- Directory of stubs source files. Stubs are organized into two groups:
  - Auto generated stubs contain all symbols that could not be resolved from within the tested scope.
    - These stubs were generated for the scope of the original translation unit.
    - ${project_loc}/common/stubs/autogenerated/auto_5c730dee.c
  - User defined stubs, for redefining in scope functions, are organized based on original source file name.
    - ${project_loc}/common/stubs/city_c_stubs.c
    - ${project_loc}/common/stubs/diptreaty_c_stubs.c
    - ${project_loc}/common/stubs/player_c_stubs.c
    - ${project_loc}/common/stubs/requirements_c_stubs.c
    - ${project_loc}/common/stubs/tile_c_stubs.c
- C/C++Test test configurations:
  - ${project_loc}/Run GNU GCC Tests with Assembly Coverage Monitoring.properties
  - ${project_loc}/Run Unit Tests (Instrument selected TU only).properties

Review of C/C++Test project setup:

C/C++Test uses its own set of project properties to understand the layout of each project. The testing and execution environments are described in "Test Configurations". Below are the project's necessary modifications from the default versions shipped within C/C++Test to suite the FreeCiv project. These modifications are supplied with the Parasoft fork of the Freeciv git project.

Project properties - C/C++Test build setting:

- Build command line – “make clean all -j8 -i COPTS=’’ CXX=${CPPTEST_SCAN} CC=${CPPTEST_SCAN} CCLD=${CPPTEST_SCAN}”
- Build working directory – “${project_loc}/”
- Dependency file(s) - “${project_loc}/Makefile”
- Compiler Family – defaults to 9.x 64bit, change as needed to suite your environment.
- Compiler options - “-DFREECIV_NDEBUG ${cpptest:original_options} -DPARASOFT_CPPTEST”

Review of initial state for the unit testing for the file ${project_loc}/common/metaknowledge.c

FreeCiv, for its original purpose, does not require rigorous safety critical testing. For this reason, we have abandoned the requirements base testing practice in favor of focusing on testing complete source level structural coverage for modified condition and decision coverage. By abandoning the requirements-based testing procedures we are assuming the original source code is correctly implemented for its designated tasks.

- Before focusing on assembly level structural coverage it is advisable to produce complete source level structural coverage.
    - The supplied Unit Tests for the file metaknowledge.c produces 100% source line and MC/DC coverage.
    - Test cases are designed using a minimal set of unique-cause + masking MC/DC parameter lists.
    - ${project_loc}/common/stubs directory contains function bodies for both missing functions and functions used to drive the flow of execution during testing.
    - Documentation on each testcase supports the validity of each row of parameters used.
- Examine modified source level MC/DC execution flow target.
    - ${project_loc}/Run Unit Tests (Instrument selected TU only).properties
    - Common C/C++Test testing configurations:
        - Execution – Symbols – Symbol sources – uncheck all options.
        - Execution – General – Unit Testing Settings – Test suite file search patterns(*) “${project_loc}/*”
- Produce a source level coverage report.
- Brief review of resultant reports to show scope of unit testing and the accumulation of coverage statistics.
- Review execution scope: test case, test suite, whole or partial project.
- Examine modified assembly execution flow target
    - ${project_loc}/Run GNU GCC Tests with Assembly Coverage Monitoring.properties
    - Review modifications to assembly acronym for project compiler
        - The same modifications as above for source level coverage
        - Execution – General – Execution Details – Target_Assembly_acronym “x86_64_gcc”
    -
    - Point out features of Asmtools accessible from C++Test UT UI
    - Cover when and when not to create discreet versions of the execution flow
    - Review modification to asm flow necessary for execution context (can this be done cleanly)
        - Cross target

        - Native
        - Debugger
    - Review switches for asm report generation
- Run asm coverage for the same scope as the original MC/DC coverage report
- Review base practice for monitoring execution and resolving errors
    - Console logging
    - File system resource locations
    - Log and results file locations for this project
    - Demo running the UT through the supplied debugger
    - Can this just be an execution flow called debug?
- Present the final report noting coverage gaps not expressed in MC/DC

Summarize the demonstration terms of DO-178C

- Define the mandate for structural machine code coverage level A
- Acknowledge the coarse buckets of machine code that have difference structural coverage when transitioning for completed MC/DC to beginning assembly coverage
    - Machine checks
    - Implicit edge case checks on conditional constructions
- Cite direct references to the guiding principal of DO-178 source level MC/DC is constructed as the minimal set of unique-cause + masking MC/DC and masking MC/DC independent pairs to show source level structural coverage