CO-LaN Test Suite

Malcolm Woodman, M R Woodman Consulting / UK

Overview

- Feedback from end user testing survey
- Test Suite high level design
- Test Suite demo
- Feedback & discussion

Feedback from Testing Survey (1)

- 9 full responses, 1 partial
- Everyone does it differently
  - including testing of CAPE-OPEN implementations
- Software testing in general
  - Mostly a mixture of automated and manual
  - Nobody does it all automated, very few all manual
- Testing of CAPE-OPEN implementations
  - Manual
  - Infrequent, if at all
  - Not clear if testing that is done is on development builds, or on clean install of final release candidate

Feedback from Testing Survey (2)

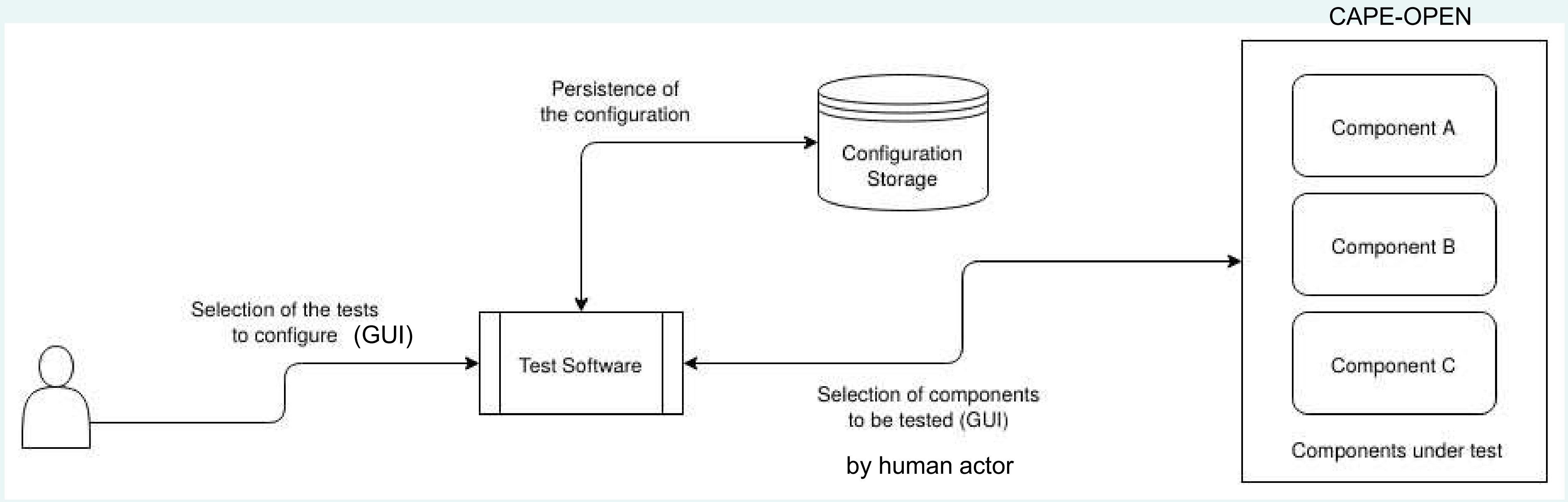
- Automated 3rd party (testing) tools:
  - Many different tools, each used by a limited number of vendors
  - Google test
  - Python Unittest
  - Microsoft CodedUI
  - Appium
  - DUnitTest in Delphi
  - Jenkins
  - CruiseControl
  - FinalBuilder
- In house tools
  - Used by highest proportion of vendors
  - C++ / C# interface required
- Command line driver required

Feedback from Testing Survey (3)

- When would use test suite
  - Many would use during development cycle
    - if quick to run and can be integrated
  - Otherwise before beta or final release only
- Output report
  - Needs to be parseable, e.g. NUnit, XUnit or JUNit
- Certification
  - Mostly “Yes”
  - For currently maintained implementations
  - But need a business case

Test Suite High Level Design

- Initial design is for Process Modelling Components (PMCs) only
  - Extension to Environments (PME) will follow


CAPE-OPEN
Persistence of the configuration
Configuration Storage
Component A
Component B
Component C
Components under test
Selection of the tests to configure (GUI)
Test Software
Selection of components to be tested (GUI)
by human actor

Running the Test Suite

- End-user configures:
  - The components to be tested
  - The selection of tests to run
- End-user runs the Test Suite to execute the selected tests on the required components
- The test software provides the persistence of the configuration for reuse

Architecture

- There are no test specific interfaces to be implemented in the component under test
  - The component is tested "as-is"
- The Test Suite is built using CAPE-OPEN interfaces
- The Test Suite is built using the COBIA middleware
  - Will therefore support all of
    - CAPE-OPEN v1.2
    - CAPE-OPEN v1.1 (via COMBIA)
    - CAPE-OPEN v1.0 (via COMBIA)
  - But NOT Thermo v1.0
    - Deprecated!

Integration into Automated Testing

- The proposed design splits the Test Suite into:
  - The Test Host – the user interface
  - The Test Engine – which handles the execution of the tests
  - Communication between the Host and Engine is handled via the “Test Engine Interface”, which is open and extensible
- The design therefore allows integration into any of the 3rd party tools mentioned earlier, by developing a new Test Host
  - But little commonality on tools, so which ones are important?
- Integration into in-house tools
  - C++ already possible
  - C# in the future
    - There is currently no .NET binding for COBIA

Hierarchy of Tests

- Compliance Tests
  - E.g. for Thermo PMC or Unit PMC
  - All the tests that need to be run in order for a request for certification to be submitted
  - Note, all the tests may not need to be passed successfully, e.g. if any of them are irrelevant for the specific PMC under test
- Test Categories
  - Groups of tests with the Compliance Tests that need
    - Similar setup
    - Similar data
- Tests
  - The individual tests

Provision of Tests

- CO-LaN provided tests for compliancy
  - Defined by the relevant SIG
    - e.g. Thermo SIG for Thermo PMC
  - Will either succeed or fail with an error message
  - Test Suite users will have no direct access to the internal details of the tests from within the Test Suite, only
    - Interface definition,
    - Configuration requirements
    - Success/failure information.
- Software Developer tests
  - Any developer will be able to add additional tests
    - Register developer specific component, implementing the ICapeTestCategory interface
    - Design for defining the details of the tests themselves is still to be determined

Reports

- The Test Suite provides a programmatic interface, which allows any Test Host to process test messages, test failures and passed tests as necessary
- Currently results of tests are reported by:
  - A simple text output, convenient for
    - reading it in the console
    - continuous integration tests
  - An JSON format text file
- Other formats can be added in the future if required
- Note that the prototype does not protect the results file in any way
  - Can be edited, thus invalidating the results
  - Future discussion: does it need to be protected?

Test Suite Prototype - Demo

- Current Status:
  - Command line interface
  - Allows testing of Property Packages
    - but not a Property Package Manager
  - Basic tests have been implemented
    - No example of providing data for a test
    - Not the full set of tests for testing compliancy
  - Persistence has been implemented

- Live demo

Test Suite Prototype – Next Steps

- In 2021:
  - Implement further tests:
    - ICapeThermoCompound
    - Implement minimal editing window, e.g.
      - Set temperature for temperature dependent properties
  - Complete any other outstanding features necessary to demonstrate the full workflow and functionality
  - Provide to selected / volunteer CO-LaN members for review
- In 2022:
  - Extend to Property Package Manager
  - Allow additional Software Developer tests
  - Thermo SIG to define compliancy tests
  - Implement all tests defined by Thermo SIG
  - Modify design/implementation based on CO-LaN member review

Acknowledgements

- The CO-LaN Software Development Contractors
  - Marcus Bruno
  - Loic d’Anterroches
  - Jasper van Baten

Feedback & Discussion

- Feedback on design & current prototype
- Is this what you were expecting?
  - If not, what should be different?
- Volunteers to review prototype when it is available?
- ……..