Verify and automate CAPE-OPEN software compliance

Laurent TESTARD
HALIAS Technologies
www.halias.fr
Company overview

• Creation date : 2006

• 4 employees in Meylan (near Grenoble)

• Access to a network of sub contractors and field experts

• « Pedigree » :
  – « Jeune Entreprise Innovante »
  – French Ministry of Research (MESR) expertise
  – Training center
  – OSEO « GO-Innovation 2011 »

• More than 25 production and R&D sites in the world
Target audience and major issues

• Software editors
  – Economic efficiency in the development process
  – Better reliability of software
    → lower maintenance costs

• End users
  – Lower adoption costs (acceptance tests)
  – Better reliability of software
    → limit impacts of software bugs on operations

• CO-LaN
  – Better software quality → wider acceptance of the standard
  – Service to the community (editors and end users)

Better tests, more tests, faster tests!
Testing activities from an editor point of view
From the end-user point of view

• Software acceptance test activities
Standard compliance checking!

- Effort not regularly spanned over time
- Work can be (very) important on peak activities
- Complexity (combinatory: PME + PMC + OS + platform)
Enhancement of an existing testing process

• Systematization of existing tests (all targets concerned)
• Efficiency of tests (all targets concerned)

• Some work directions:
  – Automate tests (static and dynamic)
  – Lower investment costs (tools, frameworks, ...)
  – Lower update costs (increase of test volumes)
  – Measure the tests efficiency (coverage...)

10/11/2012 HALIAS Technologies - www.halias.fr 7
Static testing

- Principle: based on source code (white box testing)
  - Unit tests
  - Static code checks
  - Quality tests

- Metrics:
  - Source code coverage
  - List of non-functional checkings (QA, memory, ...)

- Benefits, costs, and limitations
Example : unit tests

- Co-development of code and unit tests
- The « xUnit paradigm » : cppUnit, jUnit, Nunit, ...
- Fixtures, test cases and test suites
- Assertions (value > 10, ...) for simple tests
- Commonly used languages
  - C++ : CppUnit, C++ Mocking framework, ...
  - Fortran : Fruit (never used...)
  - Java : Junit
  - C# :
    - Visual Studio Team Tests,
    - Nunit ? (still existing ?)
- Goal : 50-80% coverage ? Poll ?
- Error cases coverage ...

- Conclusion : these are mandatory development activities
Functional testing

- Verify the functionalities of software
  - wrt what is really expected (by users, clients, QA, ...)
  - Black Box Testing
  - Better understanding of software capabilities and limitations

- Many different methods exist:
  - GUI testing
  - Value testing
  - CAPE-OPEN Standard compliance testing

- Useful metrics:
  - Functional coverage
  - Goal: 100% nominal, error and corner cases?
CAPE-OPEN standard compliance testing: what could be checked?

- **PMEs:**
  - Standard scenario and expected behavior. Example of initialization orders?
    - Example: Initialize / Validate / Calculate
    - Precedence of SetValues / CalcEq
  - Crash detection
  - Multiple platforms and OS comparisons
  - Backward compatibility on reference flowsheets
  - Component technology testing (.NET vs DCOM/...)
  - Persistence considerations
  - Basic CAPE-OPEN related performance studies (overhead, ...)

- **PMCs**
  - Bounds of physical values, simple relations (100% for phases sums), statistical characterization ...
  - “shell” development (for a Thermo PP Server) for standard data retrieval
  - Testing native code against CAPE-OPEN wrapped code (values)
  - Component technology testing (.NET vs DCOM/...)
  - CAPE-OPEN versions cross-compatibility (1.0/1.1/...)

Benefits of an operational testing framework

1. **Wider diffusion** of the standard
   - A release of a PME will be better tested on CO aspects
   - Compatibility with earlier versions
   - Detection of regressions for more stability

2. **Efficiency of the compliance checking process**
   - Compliance checking **automation**
   - Maximum coverage (OS/versions/features)

3. **Capitalization** of compliance checking **data and methods**
   - Collaborative / participative workflows
HALIAS Proposition

• Co-develop a technical infrastructure dedicated to automated standard checking of compliant software and based on the QTS solution.
  
  – Benefits the CO-LaN
  – Benefits the CO-LaN members

• Tests results remain confidential

• Technical proposition, to initiate discussions...
Proposed solution

CO-LaN toolbox: COLTT log files analyzer, PMC shell, ...

QTS Server

Compliance reports

PME

QTS Watchdog

COLTT

Shel I

PMC

PME Editor

PMC Editor
QTS Client node infrastructure

• Provisioning of VM images containing various combinations
  – PMEs,
  – PMCs,
  – OS (supported windows : 32/64 XP/Vista/7), ...

• Automatic testing process :
  – resurrect VM,
  – launch tools,
  – analyze results,
  – kill VM

• Test data and tested software management
  – Remote secured access to the VM
  – Each member can access its VM
  – Reservation system
Conclusions

- HALIAS can provide a solution that automates standard compliance checks of CAPE-OPEN compliant tools.

- HALIAS is willing to share with the members of the CO-LaN his testing know-how.

- More kinds of tests can be achieved, any specific request can be discussed (open environment)!
  - Testing methods
  - Environments
  - Testing processes
  - Audits and specific studies

1. Better Tests: systematic capitalized tests
2. More Tests: improved coverage combinations extensibility
3. Faster tests: scalability automation
Thank you for your attention!

Any questions?

www.halias.fr
Laurent.Testard@halias.fr

57 Chemin du Vieux Chêne
38240 MEYLAN – FRANCE
Tel : +33(0) 689 065343