Verify and automate CAPE-OPEN software compliance

Laurent TESTARD

HALIAS Technologies

www.halias.fr

HALIAS TECHNOLOGIES

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

Main clients

TOTAL

Axens IFP Group Technologies

TECH advantage Groupe IFP

EDF

inria

ifp Energies nouvelles

North America
Europe
Asia
Atlantic Ocean
Africa
South America
Pacific Ocean
Indian Ocean
Australia

Target audience and major issues

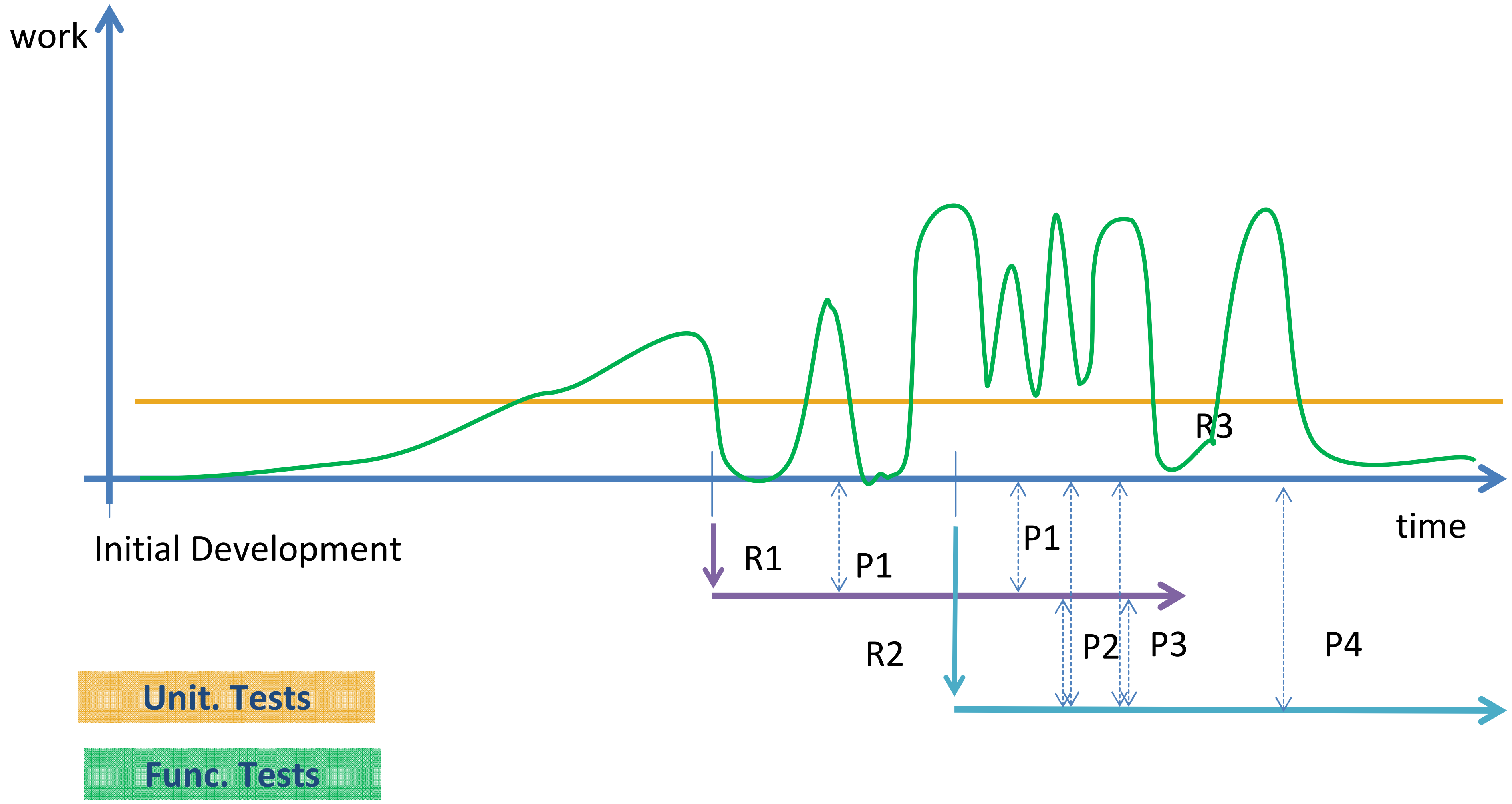
- Software editors
  - Economic efficiency in the development process
  - Better reliability of software
    → lower maintenance costs

- 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)

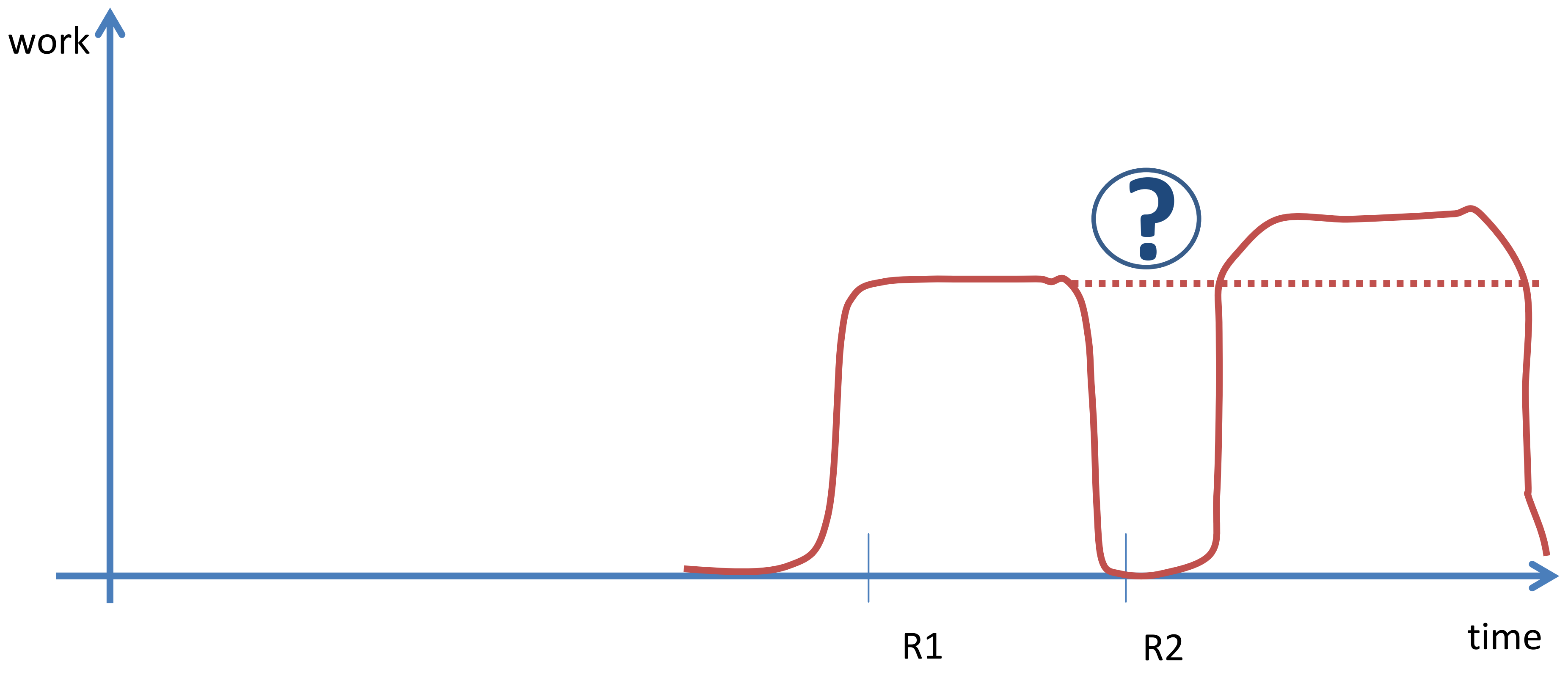
Better tests, **more** tests, **faster** tests !

Testing activities from an editor point of view



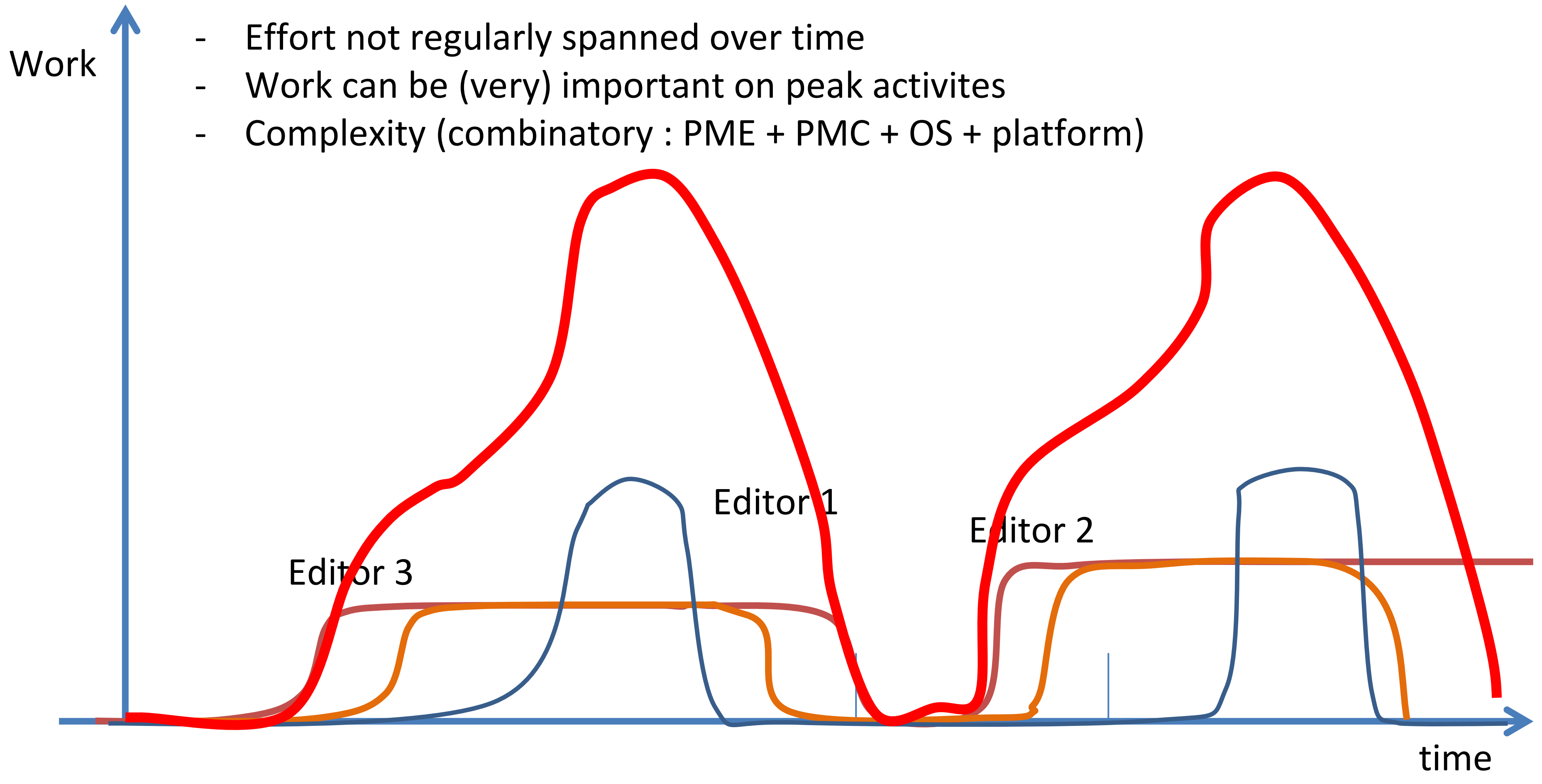
From the end-user point of view

- Software acceptance test activities



Standard compliance checking !

- Effort not regularly spanned over time
- Work can be (very) important on peak activites
- Complexity (combinatory : PME + PMC + OS + platform)



HALIAS TECHNOLOGIES

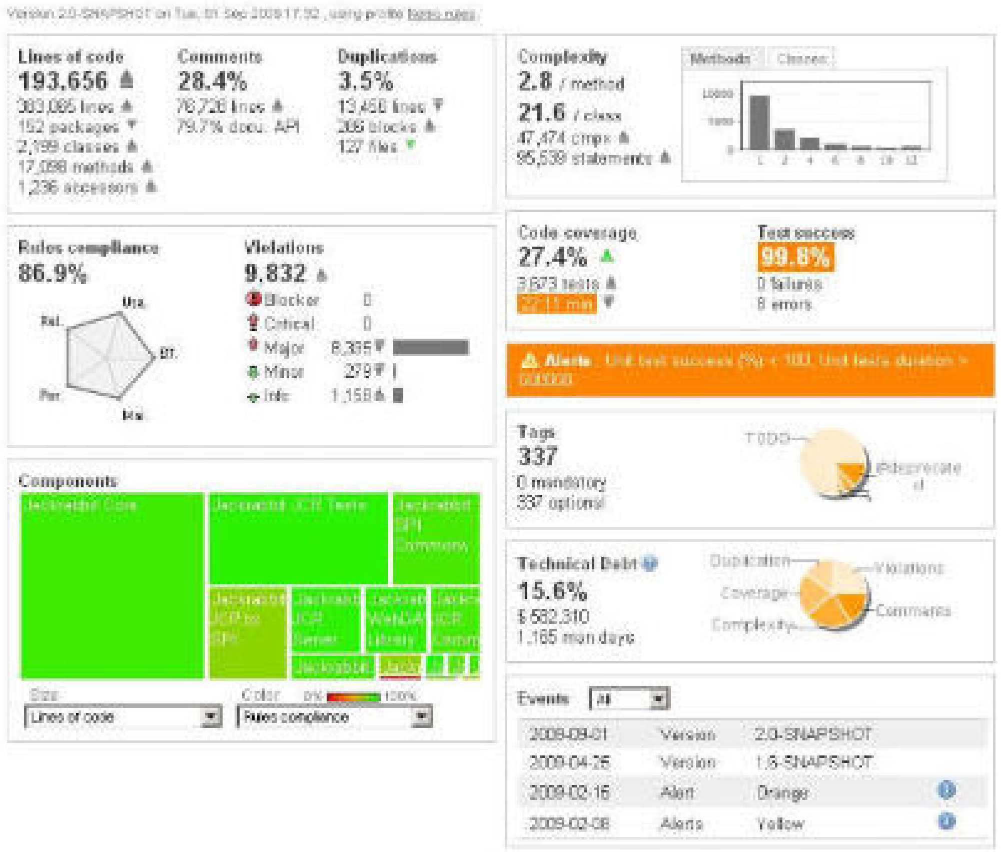
Enhancement of an existing testing process

- 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…)

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



Version 2.0-SNAPSHOT on Tue, 01 Sep 2009 17:32 , using profile [illegible]

Lines of code
193,656 ▲
383,085 lines ▲
152 packages ▼
2,199 classes ▲
17,098 methods ▲
1,236 accessors ▲

Comments
28.4%
76,728 lines ▲
79.7% docu. API

Duplications
3.5%
13,458 lines ▼
286 blocks ▲
127 files ▼

Complexity
2.8 / method
21.6 / class
47,474 cmpx ▲
95,539 statements ▲

Rules compliance
86.9%

Violations
9,832 ▲

Blocker	0	
Critical	0	
Major	8,395 ▼	▬
Minor	279 ▼	▏
Info	1,158 ▲	▮

Code coverage
27.4% ▲
3,673 tests ▲
22:11 min ▼

Test success
99.8%
0 failures
8 errors

⚠ **Alerts** : Unit test success (%) < 100, Unit tests duration > 600000

Tags
337
0 mandatory
337 optional

Components

Size: Lines of code
Color 0% 100%: Rules compliance

Technical Debt
15.6%
$ 582,310
1,165 man days

Events All

2009-09-01	Version	2.0-SNAPSHOT
2009-04-25	Version	1.6-SNAPSHOT
2009-02-16	Alert	Orange
2009-02-08	Alerts	Yellow

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 …

```
TEST_F(QueueTest, IsEmptyInitially) {
  EXPECT_EQ(0, q0_.size());
}

TEST_F(QueueTest, DequeueWorks) {
  int* n = q0_.Dequeue();
  EXPECT_EQ(NULL, n);

  n = q1_.Dequeue();
  ASSERT_TRUE(n != NULL);
  EXPECT_EQ(1, *n);
  EXPECT_EQ(0, q1_.size());
  delete n;

  n = q2_.Dequeue();
  ASSERT_TRUE(n != NULL);
  EXPECT_EQ(2, *n);
  EXPECT_EQ(1, q2_.size());
  delete n;
}
```

- 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

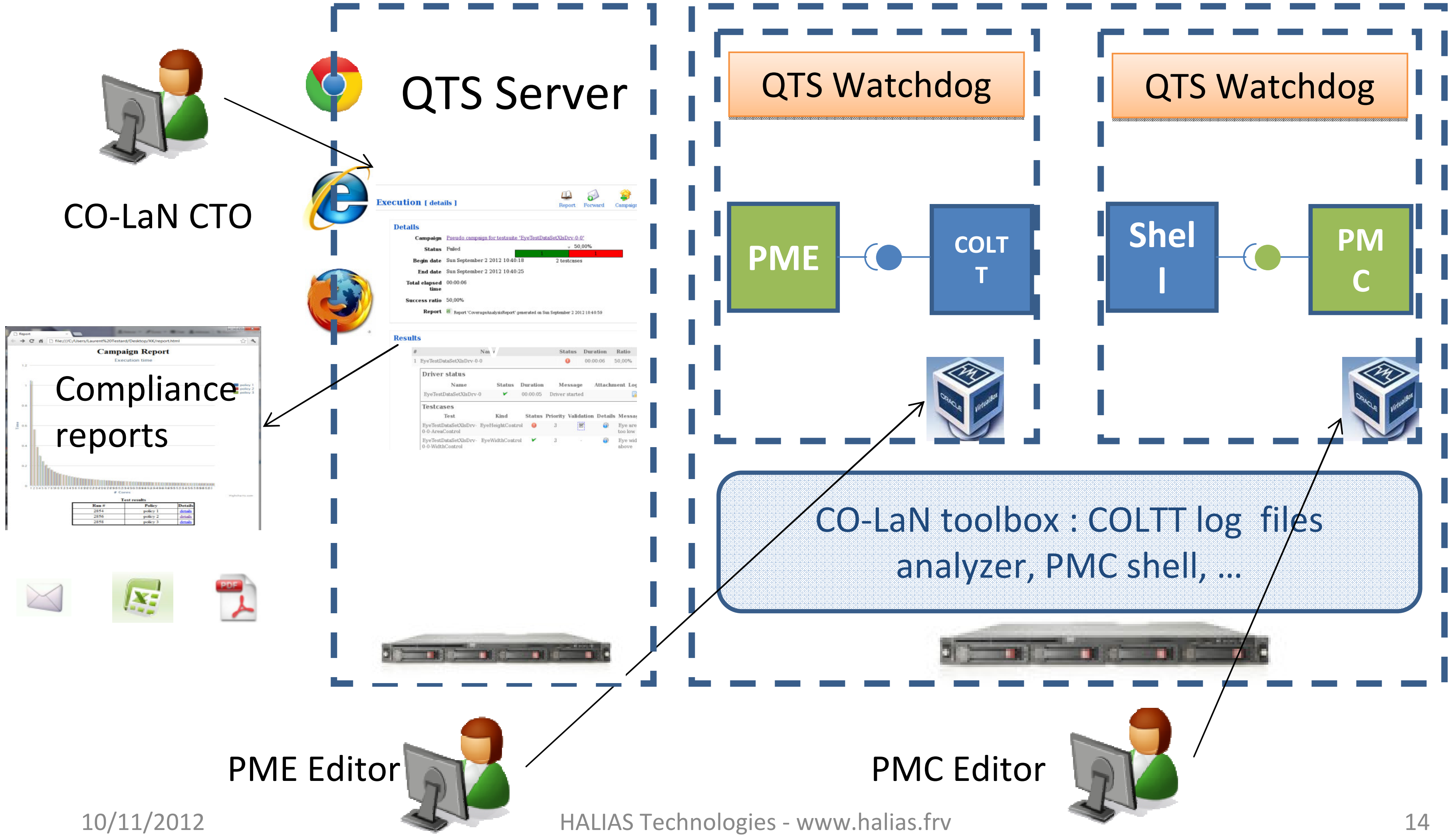
HALIAS Proposition

- **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

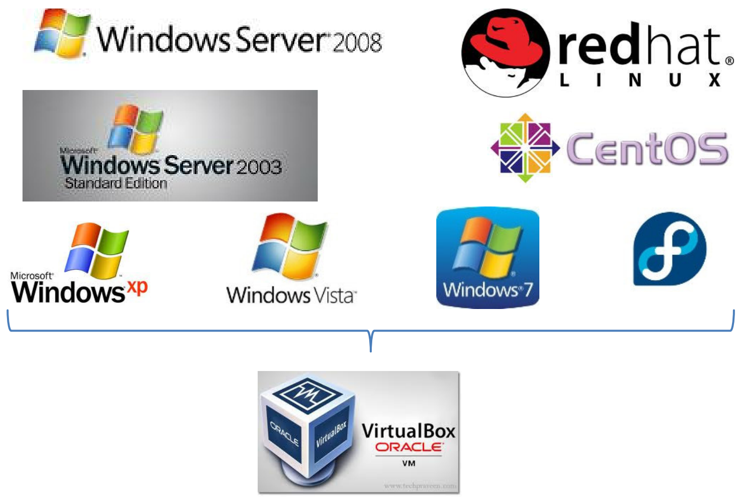
- Technical proposition, to initiate discussions…

Proposed solution



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 ?

HALIAS TECHNOLOGIES

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

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