Common Approach to Automated Acceptance Testing for Carrier Grade Telecom Platforms

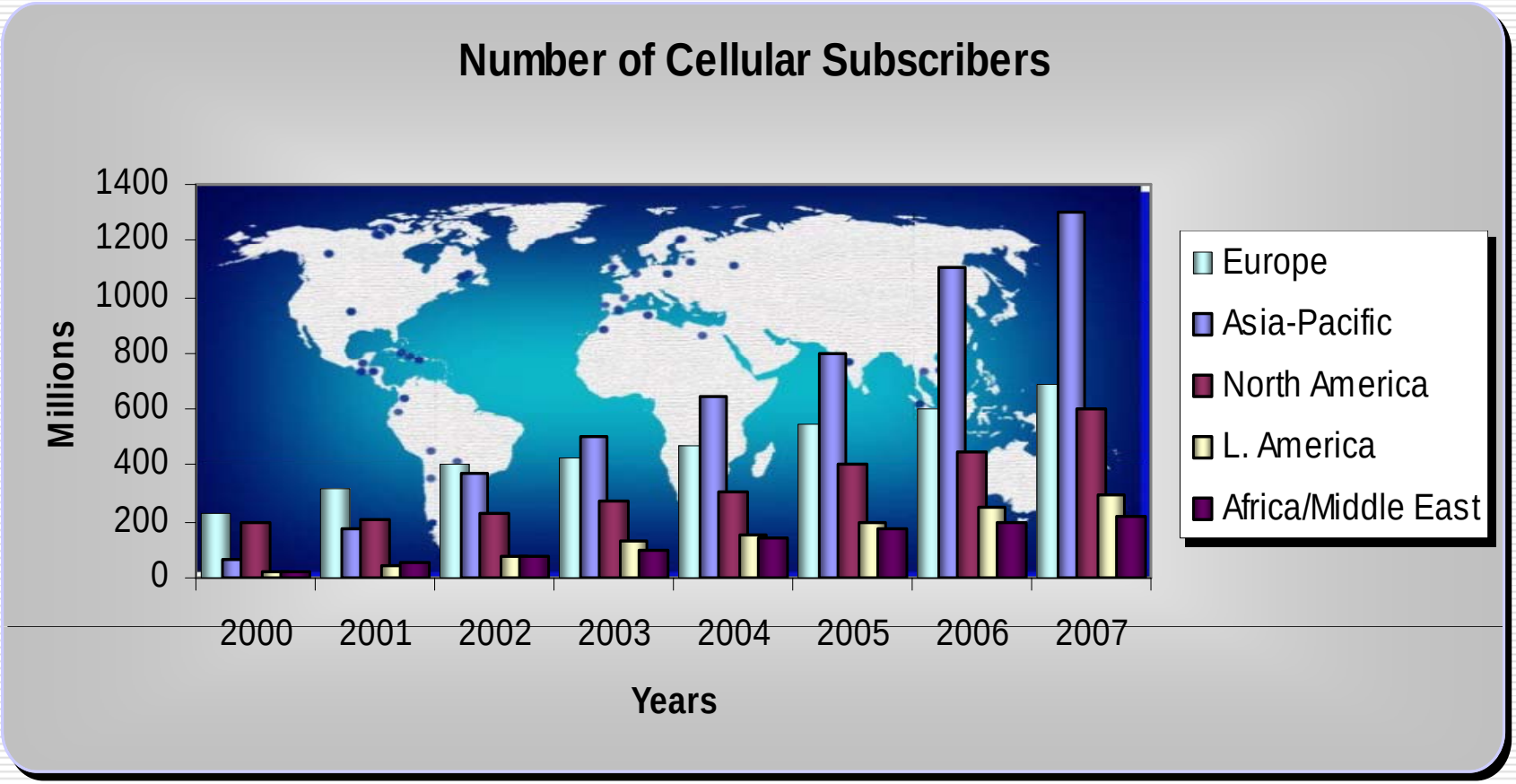
SEC(R) 2007

Vasily Linkov
Andrey Danilov
Boris Lyubimov

Content

- Growing demand for Carrier Grade Systems
- Standardization
  - Hardware
  - Operation Systems
  - Middleware and Interfaces
- Acceptance testing
  - State of the art
  - Goal
  - Scope
  - Problems
- Acceptance Testing Environment:
  - Architecture
  - Elements Description
  - Test cases
- Summary

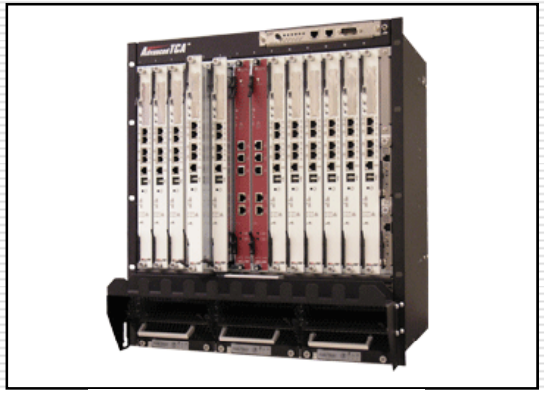
Growing demand for High Performance



High Performance & Availability Hardware

MXP3x11

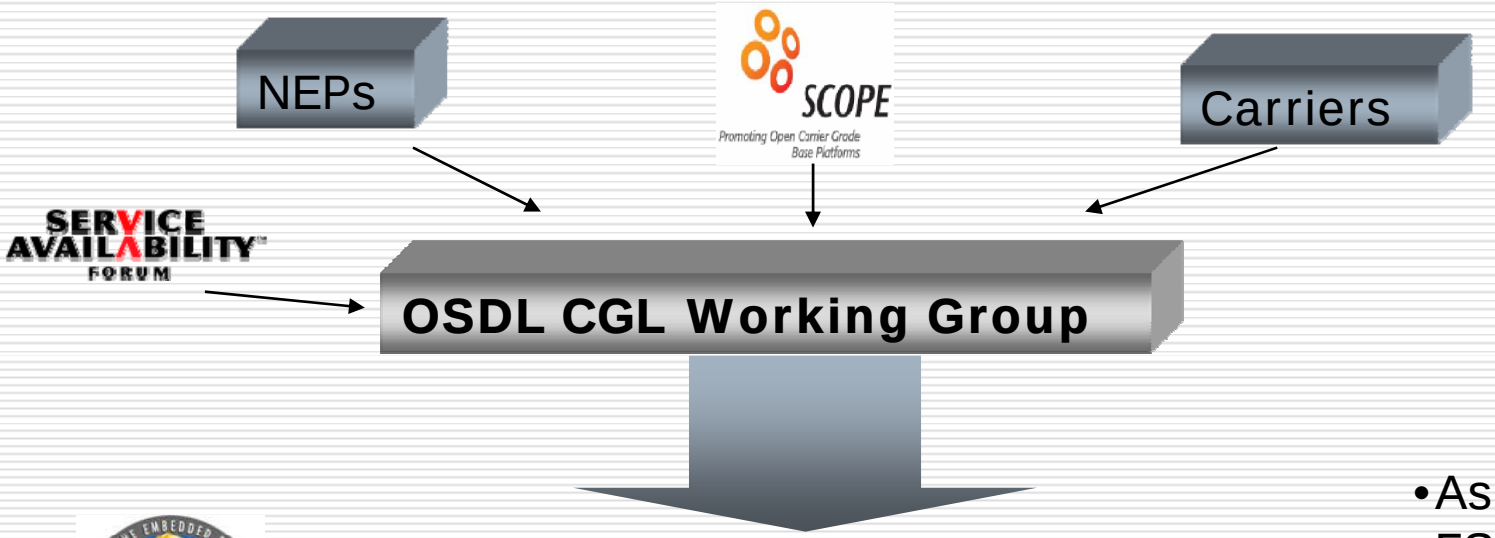
Netra CT820



AdvancedTCA®

- Standardized specifications:
  - PICMG (PCI Industrial Computer Manufacturers Group)
  - IPMI (Intelligent Platform Management Interface)

Carrier Grade Linux

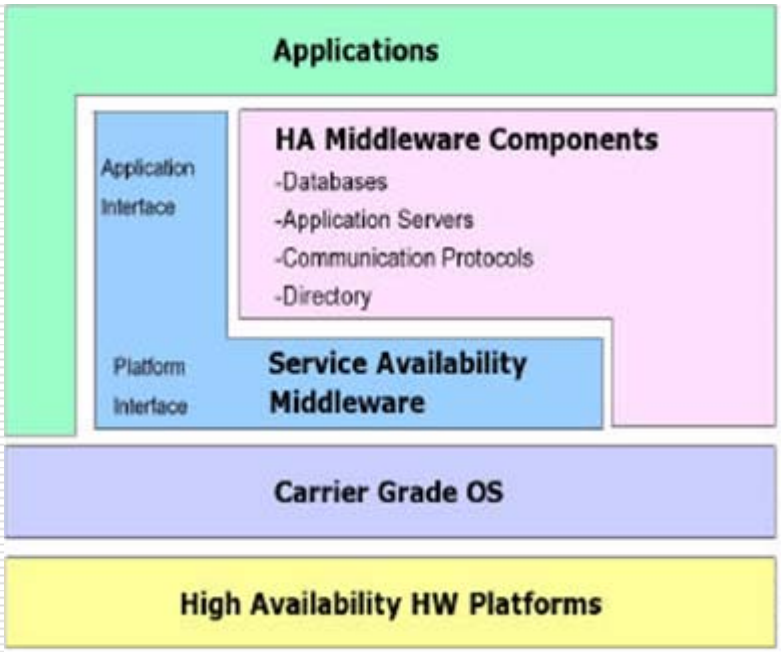


- Asianux
- FSMLabs
- MontaVista
- Novell
- TimeSys
- Wind River
- Debian

Middleware & Interfaces Standards

SERVICE AVAILABILITY™ FORUM

- Hardware Platform Interface (HPI)
- Application Interface Specification (AIS)



Acceptance Testing of CG Systems (1/2)

- Functional Acceptance testing scope
  - Expose to customer such essential features of CG Systems as:
    - *Switchovers*
    - *Graceful shutting down*
    - *Parallel or rolling software upgrades*
    - *Roll backs*
    - etc.

Acceptance Testing of CG Systems (2/2)

- Elements Acceptance Testing Scope
  - CPU
  - Memory (RAM, HDD, Flash modules)
  - *Serial and Ethernet ports,*
  - *BIOS versions*
  - *IPMI busses,*
  - *Backplane network*
  - *Specific protocols as HSSI, T1, V35 etc.*

Acceptance Testing External regulations

- Besides customers requirements in many countries government and other public organization apply strong regulations on computer equipment covering such aspects as:
  - Electromagnetic Interference and Electronic design verification. in USA it is FCC EMI regulations;
  - In Japan VCCI organization covers the same questions with CISPR 22 recommendations;
  - Canada has the Industry Canada ICES-003 standard.

Complicated procedure of compliance verification involving placing system being tested under very specific physical and environmental condition. In case of testing under influence of Electromagnetic field manual performing of testing is impossible.

Acceptance Testing (State of the art)

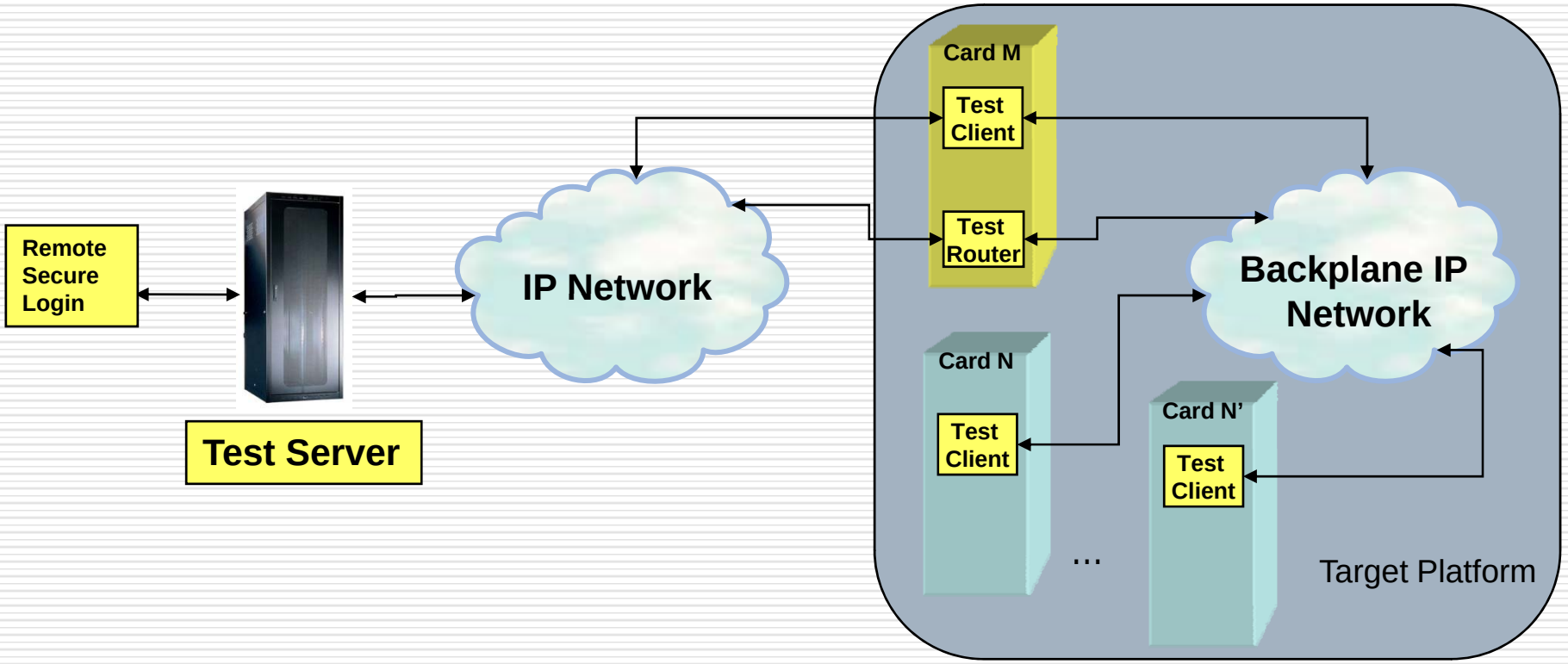
- **Linux Kernel Scalable Test Platform**
  - OSDL's Scalable Test Platform provides a test framework for Linux kernel build performance, scalability and other tests.
- **Linux Test Project**
  - Linux Test Project (LTP) is an open source project with a goal to deliver test suites to the open source community that validate the reliability, robustness, and stability of Linux.
- **SAF test**
  - Open source project providing conformance test suites for all SAF specifications, including AIS and HPI spec.
- **Other**
  - Specific frameworks exit for ATCA platforms testing, IPMI testing etc.

Most of the existing solutions are covering just small part of acceptance testing scope and usually excessively deep for acceptance testing purpose.

Common approach to Acceptance Testing benefits

- Leverage existing trends to standardization of CG Systems
- Addresses issues with automation of acceptance testing
- Gives wide and comprehensive approach to both system and element levels of testing

System Architecture



- Common ATP Diagnostic element

Test Server

- Programs and scripts upload
- Test suites and test cases execution
- Setup phase
- Clean phase
- Test suites configuration
- Selection of platform configuration for testing
- Logs Collection:
  - Execution log is intended to display status of the test suites execution.
  - Statistic log is intended to display statistic information about the test suites execution. This statistics is based on the execution log.
  - Test case log is intended to log detailed information about the test case execution. The test case log is created for each test case.
  - Event log is intended to log information about Test Server operation

Test Client

- Executes test cases and reports results to Test server
- Supports the failover testing and makes preparations for failover basing on information from test cases.
- Notifies Test Server about planned failure event coming to give test server opportunity to load to the board necessary test components as soon as board is back to the active state.

Portability or platform independence. Avoiding use of platform specific APIs helps to use test environment without changes on different CGL based systems.

Test Router

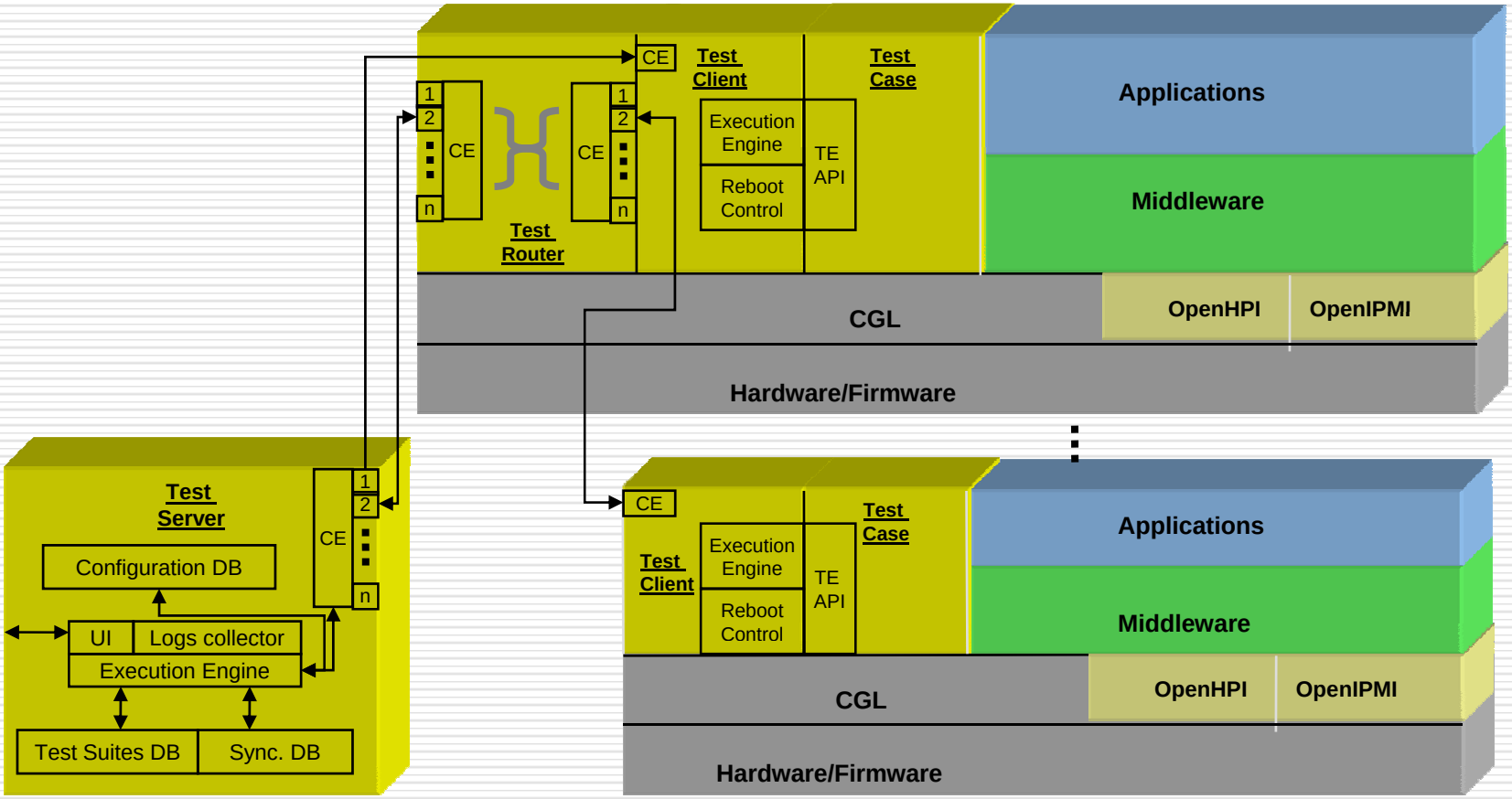
- Provides connection between Test Server and Test Client if they work in different IP networks.
- Transfers test tasks on platform elements intentionally hidden from direct access from the external network and pass testing results back to the test server.

Removing necessity of manually connect test environment to the each particular payload board and provides basis for remote testing with the same ease as local one.

User Interface

- Allows to:
  - Control test execution parameters such as stop conditions, time of execution, system configuration, and cycle mode of work.
  - Chose the scenario of testing (test suites).
  - Observe logs, which were gathered during testing.
  - Control testing status.

Software Architecture



Test case configuration file (transfer section)

```
transfer
    {
    <target_1> => "<source> <destination>",
    …
    <target_n> => "<source> <destination>"
    };
```

- Transfer section specifies which files to copy to target machines for test case.

Test case configuration file (setup section)

```
setup
    {
    <target_1> => "<script>",
    ...
    <target_n> => "<script>"
    };
```

- This section specifies what to run before the test case itself to perform appropriate setup actions like running supportive processes, creating files etc.

Test case configuration file (run/executable section)

```
run <test_name>, {
<target_1> => "<script>",
...
<target_n> => "<script>"
};
```

- This section specifies the actions needed to run the test case itself. The tasks are executed in parallel for different targets.

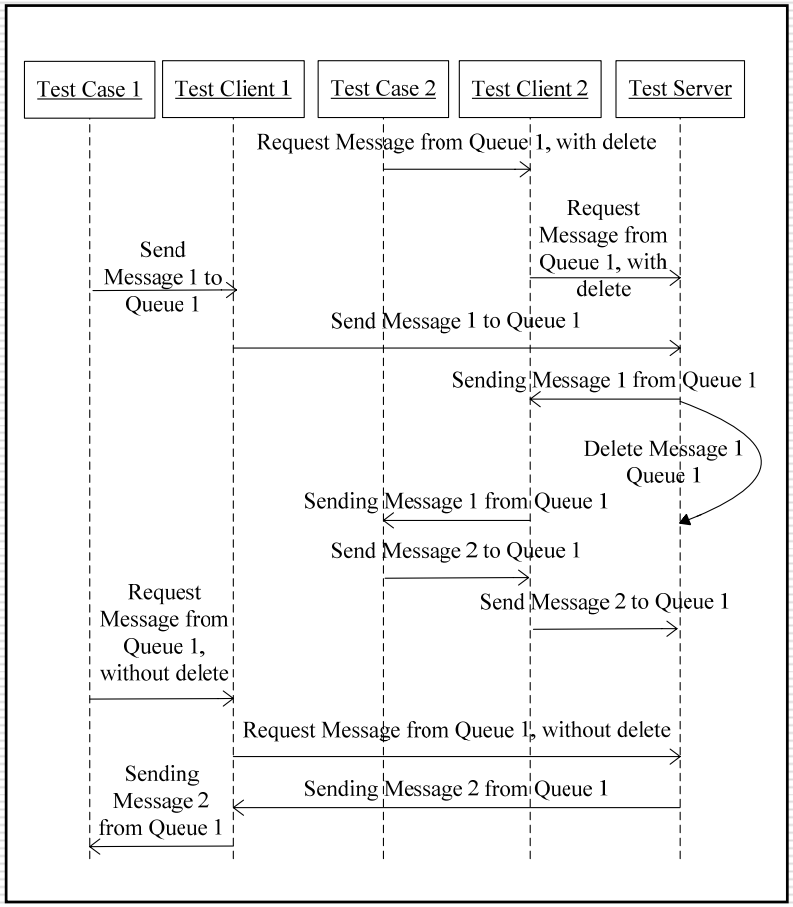
Test case configuration file (cleanup section)

```
clear {
<target_1> => "<script>",
…
<target_n> => "<script>"
};
```

- Cleanup section specifies the actions needed to perform the cleanup after the test run.

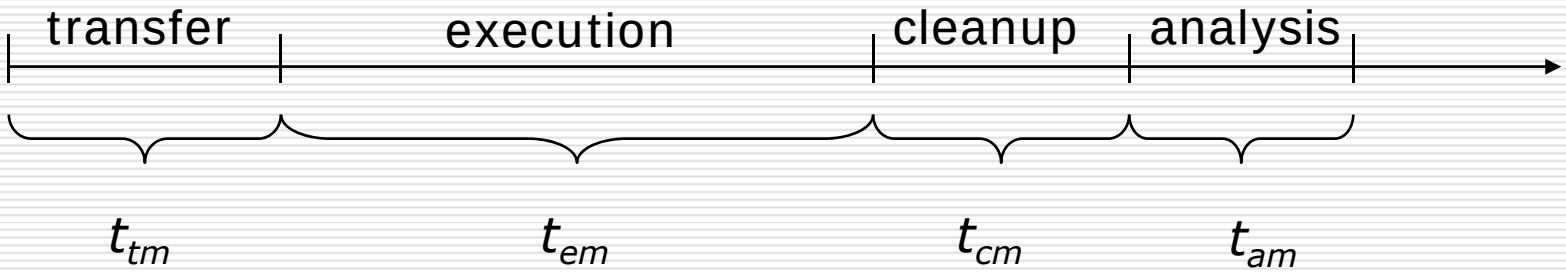
Test case

- Test case on the target board is usually C/C++ based program using specific TE APIs:
  - logs gathering
  - execution control options for failover and result announcement
  - Blackboard messaging (see chart)

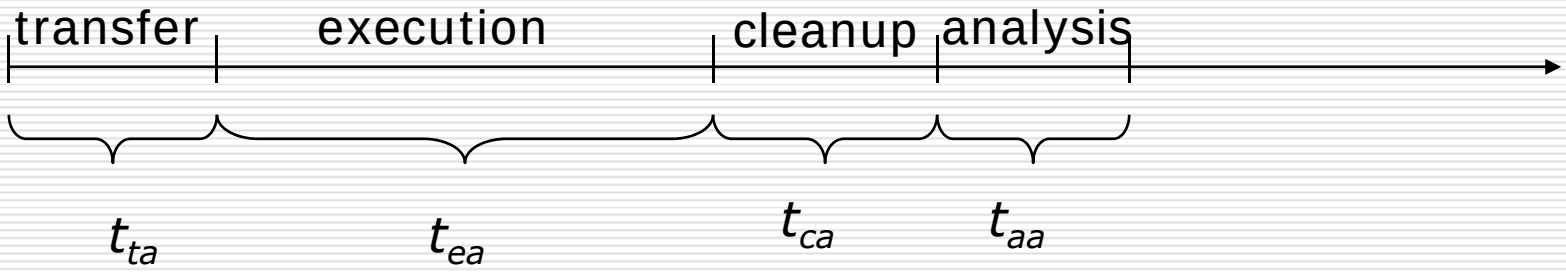


Benefits (1/2)

Manual mode

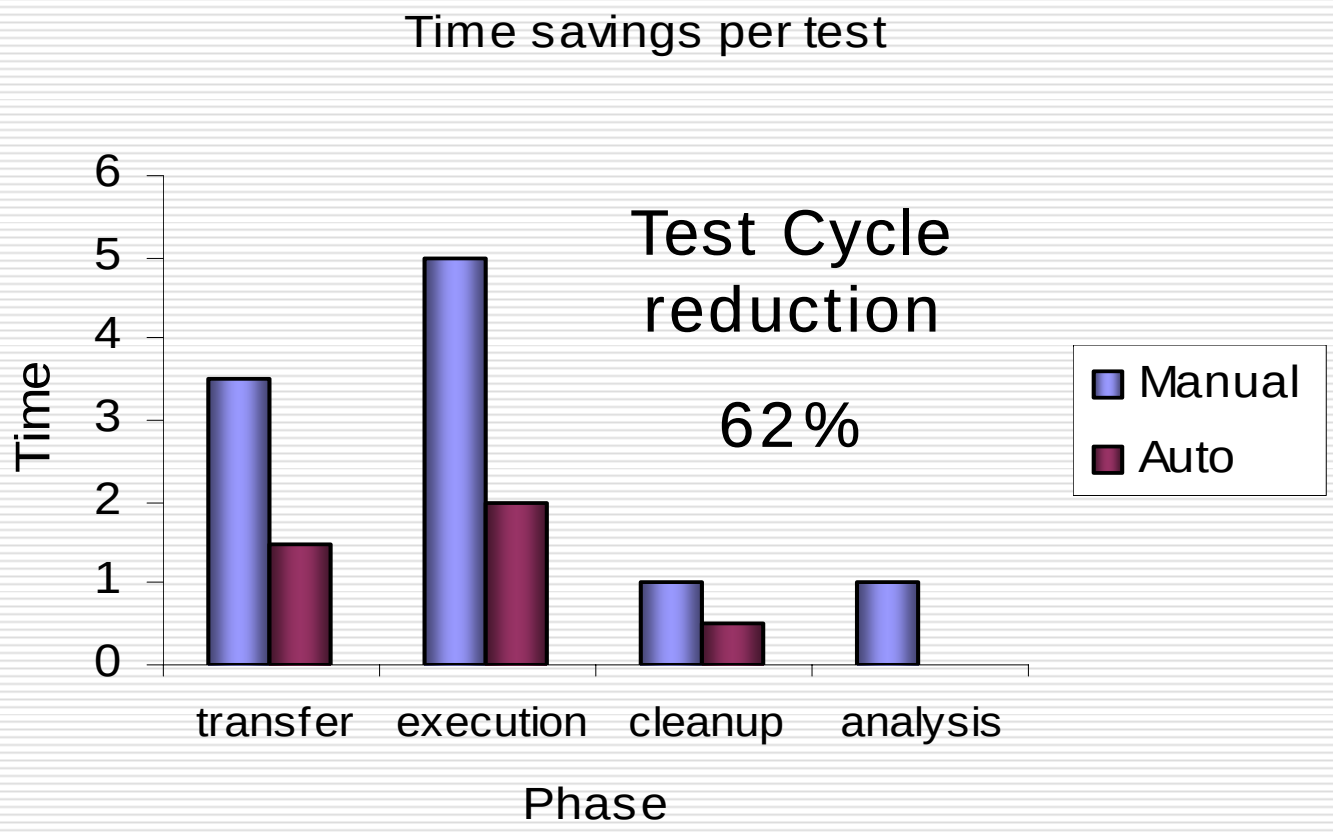


Automated mode



$$\Delta t = N_{cases}\, N_{clients}((t_{tm} - t_{ta}) + (t_{em} - t_{ea}) + (t_{cm} - t_{ca}) + (t_{am} - t_{aa}))$$

Benefits (2/2)


Time savings per test
Time
6
5
4
3
2
1
0
Test Cycle reduction
62%
Manual
Auto
transfer
execution
cleanup
analysis
Phase

Summary

- This approach is proved with three years experience of development and using of similar system.
- Helps to cover most of the critical requirements to the acceptance testing procedure and creates many competitive advantages including:
  - Commonality.
  - Work with complex network topology.
  - Extendibility.
- Decreases efforts to perform acceptance testing in comparison to the traditional manual way
- Saves time and resources for development new unique solution for new platforms.

Thank You!