

Unified functional/load test architecture and methodology using TTCN-3

TELEFÓNICA I+D
Date: June 1st, 2007



Index

- 01 Current Telefónica I+D testing scenario**
 - Functional/load tests
- 02 Telefónica I+D architecture**
 - Tests and subtests
 - Load test execution
 - Load test configuration
- 03 Telefónica I+D methodology**
 - SIP load test example
- 04 Suggested improvements to TTCN-3 specifications**

01 Current Telefónica I+D testing scenario

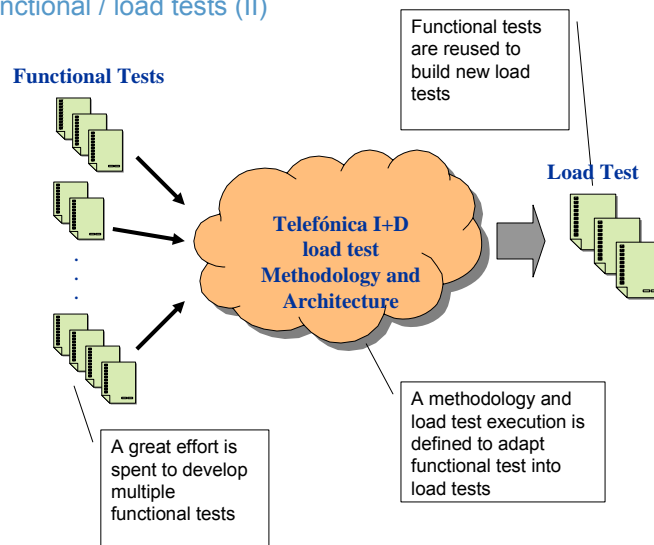
Functional / load tests (I)

- In Telefónica I+D certification processes, two main test phases can be distinguished:

- Functional test:
 - Conformance
 - Checking system requirements
- Load test:
 - Performance
 - Stress
 - Robustness

01 New Telefónica I+D testing solution

Functional / load tests (II)



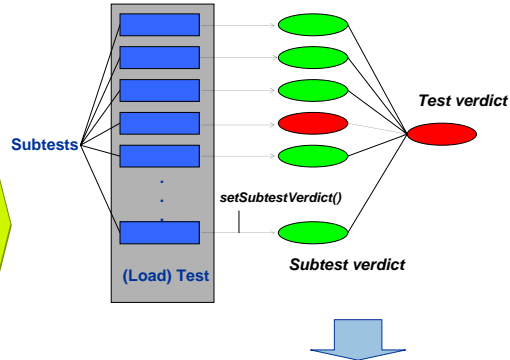
02 Telefónica I+D testing architecture

Tests and subtests (I)

TID architecture for load tests is based on the concept of subtest:

- A *subtest* is the minimum unit of test execution, to which a verdict can be assigned.

- A test (in particular, a load test) may be composed by several subtests.

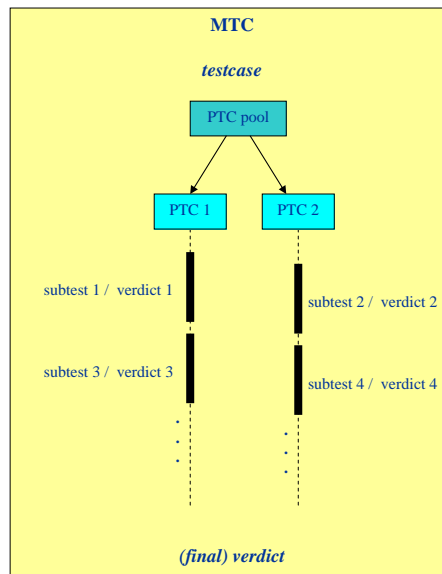


It is possible to build complex tests from existing simpler tests, retaining verdict information for them.

02 Telefónica I+D testing architecture

Tests and subtests (II)

- A good performance for load testing needs the execution of several test behaviours on the same PTC, using a PTC pool.
- Current TTCN-3 specification do not allow running more than one testcase simultaneously and associates partial verdicts to components.
- Subtests keep verdicts related to their respective behaviours.

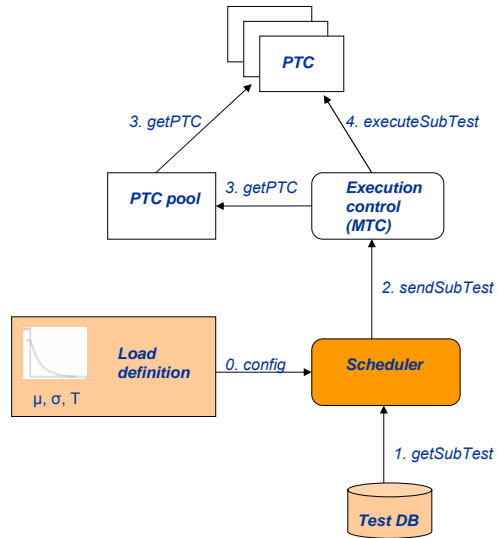


02 Telefónica I+D testing architecture

Load test execution (I)

■ Scheduler:

- Defines the instants to execute subtests, from a statistic load distribution (*load definition*)
- Controls test duration
- Gets subtest parameters from database
- In Telefónica I+D architecture Scheduler is part of the TM (*Test Manager*), but it could be included in ATS

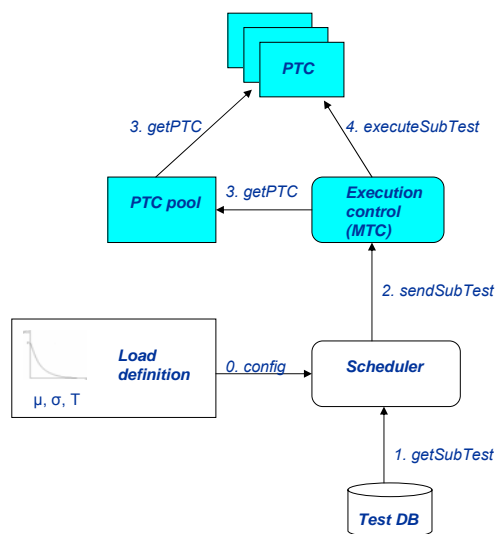


02 Telefónica I+D testing architecture

Load test execution (II)

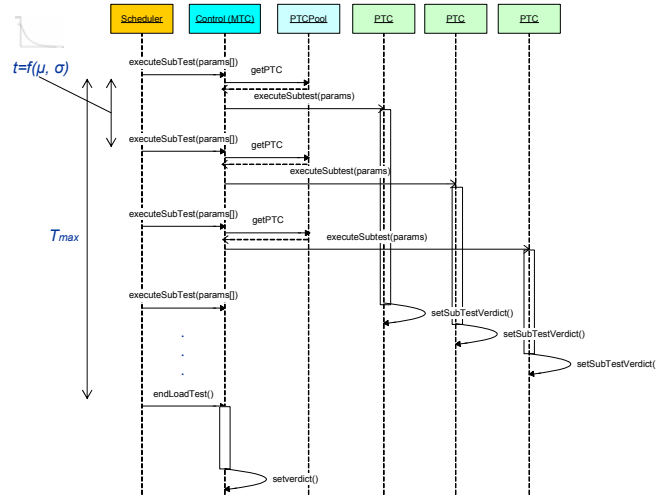
■ Execution control (MTC):

- Executes subtests in PTCs and controls their state
- Manages a PTC pool to optimize performance



02 Telefónica I+D testing architecture

Load test execution (III)



02 Telefónica I+D testing architecture

Load test configuration (I)

■ A load test is configured by:

— Test specification:

— Using a test database:

- It contains the parameters which define the subtests, i.e. it says what is going to be tested.
- It allows merging easily different types of subtests, so it is possible to model more realistic scenarios.
- A database also can be used in functional tests as a way to define sets of related tests.

— Load statistic modeling:

- Subtest execution rate (mean, variance, etc).
- Several probability density functions are possible, which can be easily configured.

02 Telefónica I+D testing architecture

Load tests configuration (II)

- Advantages:
 - Reuse of functional tests
 - Quick and simple load test configuration
 - More realistic tests
 - Increase in the number of scenarios which can be tested
 - Statistical modeling independent from test specification

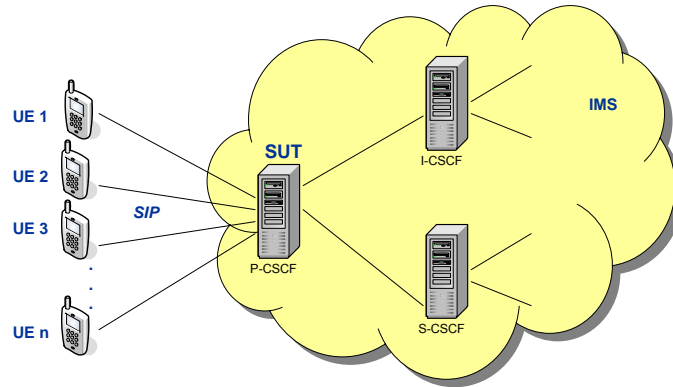
03 Telefónica I+D testing methodology

- A methodology for building load tests from functional tests is defined:
 - Specifies which conditions must be fulfilled by functional testcases and some transformation rules:
 - For example, testcases must be parameterized because several subtests will be executed concurrently with different parameters.
 - TTCN-3 behaviours could not use the order 'kill' to prevent the premature finalization of the PTC.
 - The external function 'setSubtestVerdict' must be used instead of 'setverdict', to allow keeping verdict information for subtests.
 - Etc.
 - Defines *templates* to build TTCN-3 modules.
 - Includes common modules (TTCN-3 code and adaptation libraries) to support load tests:
 - PTCs pools
 - Interaction with parameters databases
 - Subtest verdicts management

03 Telefónica I+D testing methodology

SIP load test example (I)

- Load SIP test built from functional SIP tests



03 Telefónica I+D testing methodology

SIP load test example (II)

- Load SIP test built from functional SIP tests

- From functional testcases:

- Register
- Subscription
- SIP call establishment

- Steps:

1. Make changes on original ATS:
 - Import load test support modules.
 - Modify component definitions (in which subtests are going to be executed) to make them manageable by a PTC pool.
 - Substitute 'setverdict()' by 'setSubtestVerdict()' in subtests behaviours.
2. Write load execution control module code and the main testcase.
3. Configure load test: parameters database and density probability function.

03 Telefónica I+D testing methodology

SIP load test example (III)

```
/** Simplified SIP load testcase, using a PTC pool.
 * Load execution control code.
 */
testcase loadTest(charstring px_ets_ipaddr, integer px_ets_port, charstring px_iut_ipaddr, integer px_iut_port, charstring registrar_host) runs on
SipComponent system SipInterfaces {
  trz(INFO, "Starting MTC for load tests");
  var integer subtest_counter := 0;
  // PTC pool initialization
  var Pool ptcPool := {self};
  while (not isLoadTestStoped()) {
    // Gets next subtest
    var LoadTestParams params := popSubtest();
    var SipComponent ptc;
    // Look for a free PTC from pool
    ptc := getPTCFromPool(ptcPool);
    // Executes testcase
    subtest_counter := subtest_counter + 1;
    trz(INFO, "Starting subtest " & int2str(subtest_counter));
    ptc.start(step_test_register_param(params[1].charstring, params[2].integer, params[3].charstring, params[4].integer,
    params[5].charstring, params[6].integer);
  }
  setverdict(pass);
}
```

03 Telefónica I+D testing methodology

SIP load test example (IV)

```
/* Modified functional test (SIP register). */
function step_test_register_param(charstring px_ets_ipaddr, integer px_ets_port, charstring px_iut_ipaddr, integer px_iut_port, charstring registrar_host, integer
subtest_index) runs on SipComponent {
  // Initialize
  map (self: SIPP, system: UDP1);
  ...
  // Send register
  ...
  // Timestamp for statistic generation
  timestampLoad("RegisterStart", subtestIndex);
  SIPP.send (REGISTER_Request_s_1(v_RequestUri, v_CallId, v_CSeq, v_Contact, v_From, v_To, v_Via));
  // Wait for response
  Tack.start (PX_TACK);
  alt {
    [] SIPP.receive (My_Response_REGISTER_200(v_CallId, v_CSeq)) {
      Tack.stop;
      setSubtestVerdict(pass, subtestIndex);
    }
    [] Tack.timeout { setSubtestVerdict(fail, subtestIndex); }
  }
  // Timestamp for statistic generation
  timestampLoad("RegisterEnd", subtestIndex);
  ...
  unmap (self: SIPP, system: UDP1);
}
```

03 Telefónica I+D testing methodology

SIP load test example (V)

■ Test parameter database example

px_ets_ipaddr	px_ets_port	px_iut_ipaddr	px_iut_port	registrar_host	P_PTC_1_REQUEST_LINE_E	P_PTC_1_STATUS_LIN E_R
10.95.25.64	5061	10.95.25.64	5060	10.95.25.104	-	*
10.95.25.65	5061	10.95.25.64	5060	10.95.25.104	{ method:=REGISTER_E, requestUri:{ scheme:="sip", hostPort:={ host:="ptt4.trial.net" } }, sipVersion:="SIP/2.0" }	{ sipVersion:="SIP/2.0", statusCode:=200, reasonPhrase:="OK"}
10.95.25.66	5061	10.95.25.64	5060	10.95.25.104	-	*
10.95.25.67	5061	10.95.25.64	5060	10.95.25.104	-	*
10.95.25.68	5061	10.95.25.64	5060	10.95.25.104	-	*
10.95.25.69	5061	10.95.25.64	5060	10.95.25.104	-	*

04 Possible improvements to TTCN-3 specifications

- To add the concepts of subtests and subtest verdicts
- Support for parameter databases:
 - It may be implemented within TM (Test Management system).
 - However, allowing database from ATs could improve power and flexibility of ATs.
- Improvements in PTC management:
 - PTC pools
- Improvements in log management:
 - Log filtering at TE (TTCN-3 Executable) for better performance
 - Several detail levels
- Support for statistic generation:
 - Timestamp mechanism correlated to subtests

