Scenario Care Load Testing in TTCN-3

Ji Wu

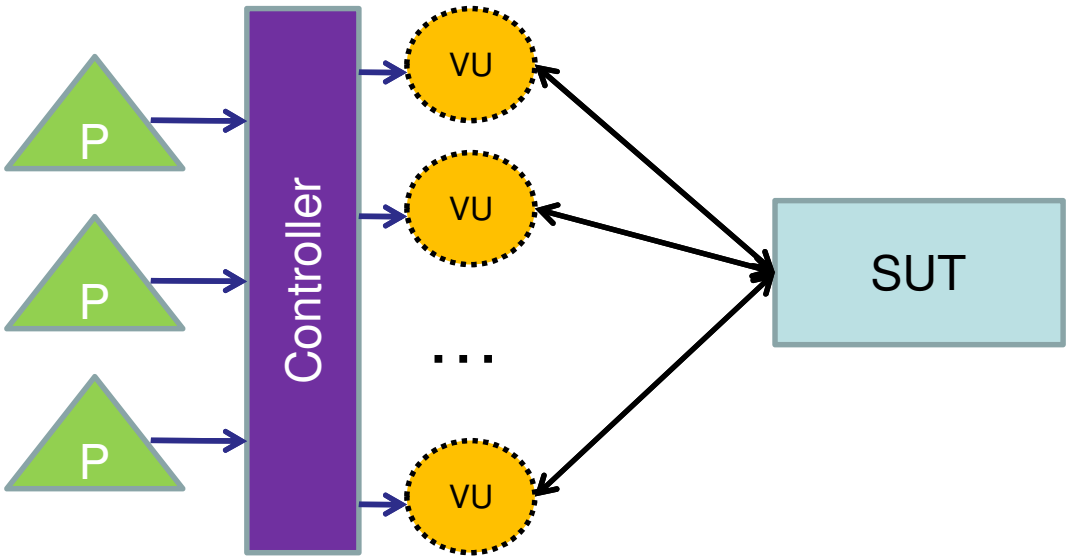
BeiHang University, China

Agenda

- Load testing in TTCN-3
- Load profile model
- Load control
- Test System Framework
- Virtual user Implementation
- Reuse existing test case
- Experiments
- Conclusion

Structure of Load testing

- Load testing is an important way to explore SUT load and the bottleneck.



A group of Virtual Users to implement the load on SUT

A Controller coordinates the Virtual Users

The coordination satisfies the given parameters

Load Testing in TTCN-3

- Load testing in TTCN-3
  - Test cases/functions for implementing load in the name of virtual users
  - Controller is part of load strategy
- Concerns on load testing
  - More virtual users
  - More varieties of user behaviors
  - More realistic load profile
- Need piles of TTCN-3 code to implement the concerns!

Load Testing in TTCN-3

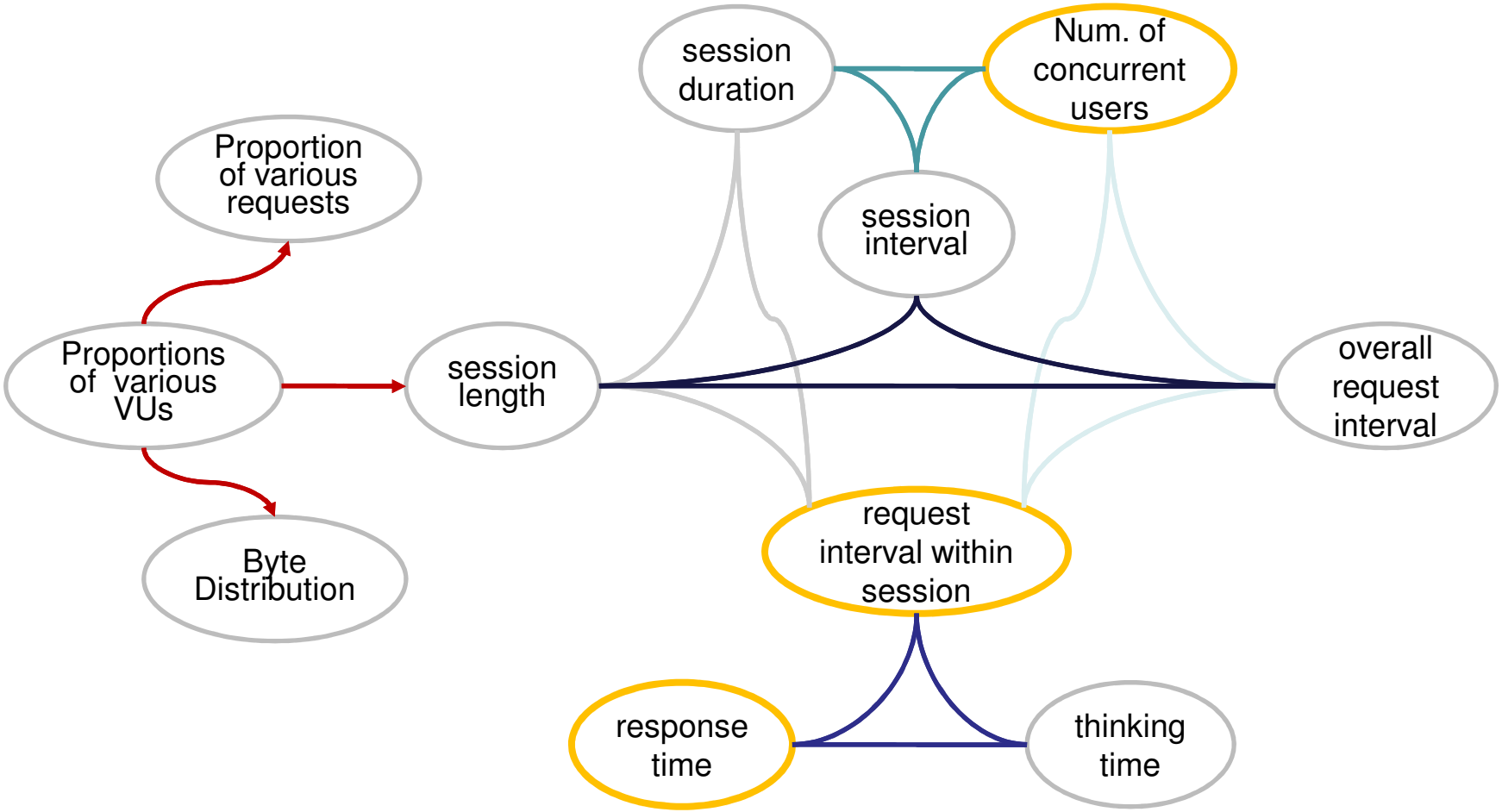
- Load testing usually goes after successful functionality testing
- The existing test cases can guarantee
  - The varieties of behaviors
  - The realistic of load profile
- Two issues come
  - How to reuse the existing test cases?
  - How to coordinate the VUs with the test cases to get good load profile?

Load Testing in TTCN-3

- How to reuse the existing test cases?
  - a question of code conversion and generation
- How to coordinate the VUs …?
  - A question of adaptive control
    - assign test cases for VUs
    - observe load profile
    - evaluate against given parameters
    - modify the assignment

Load Profile Model

Several relative metrics of load => clusters

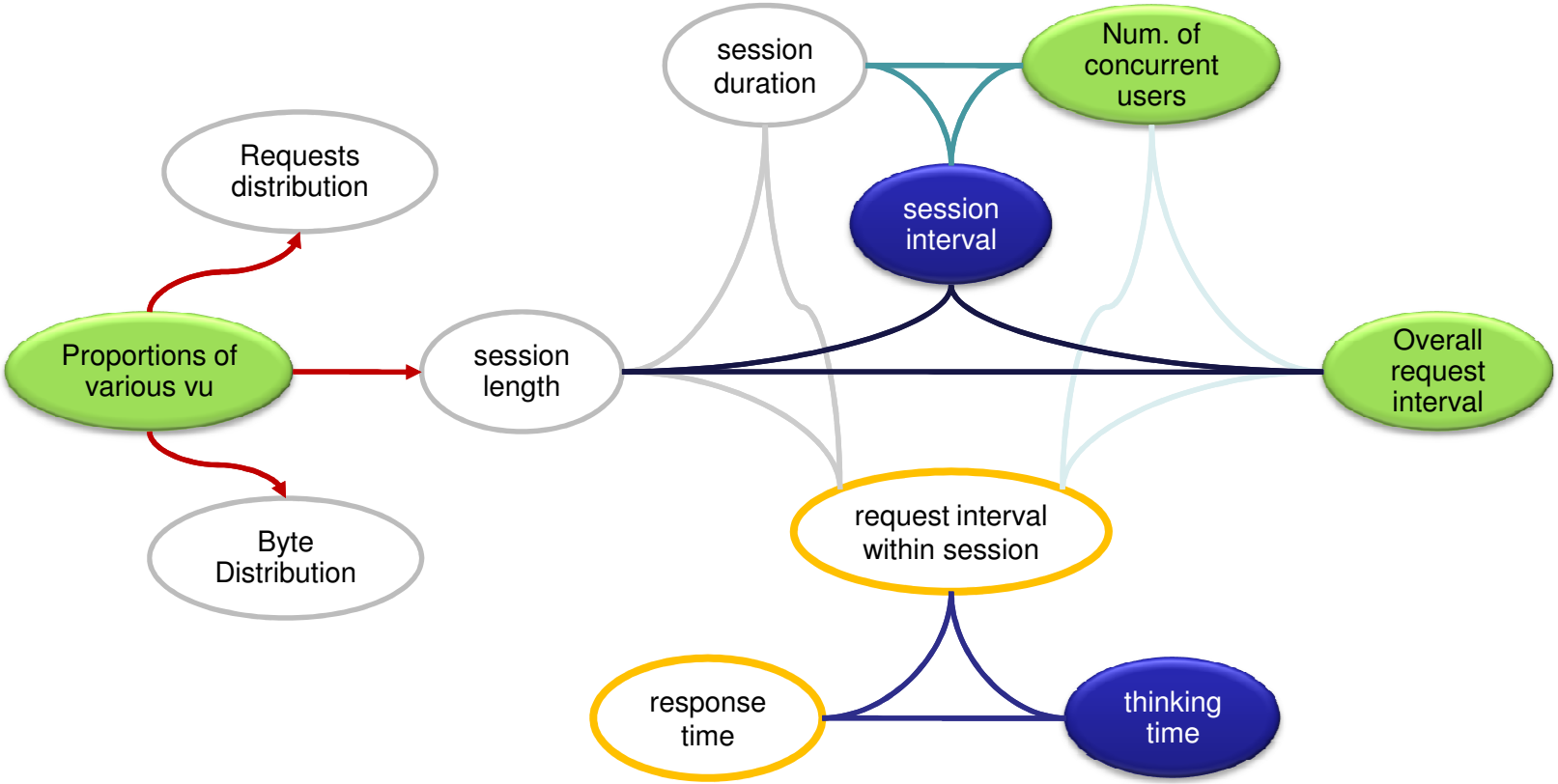


Cluster Domination Equations

session duration	=	*request interval within session*	*	*session length*
request interval overall	=	*request interval within session*	/	*number of concurrent users*
number of concurrent users	=	*session duration*	/	*session interval*
session interval	=	*overall request interval*	*	*session length*
request interval within session	=	*thinking time*	+	*response time*

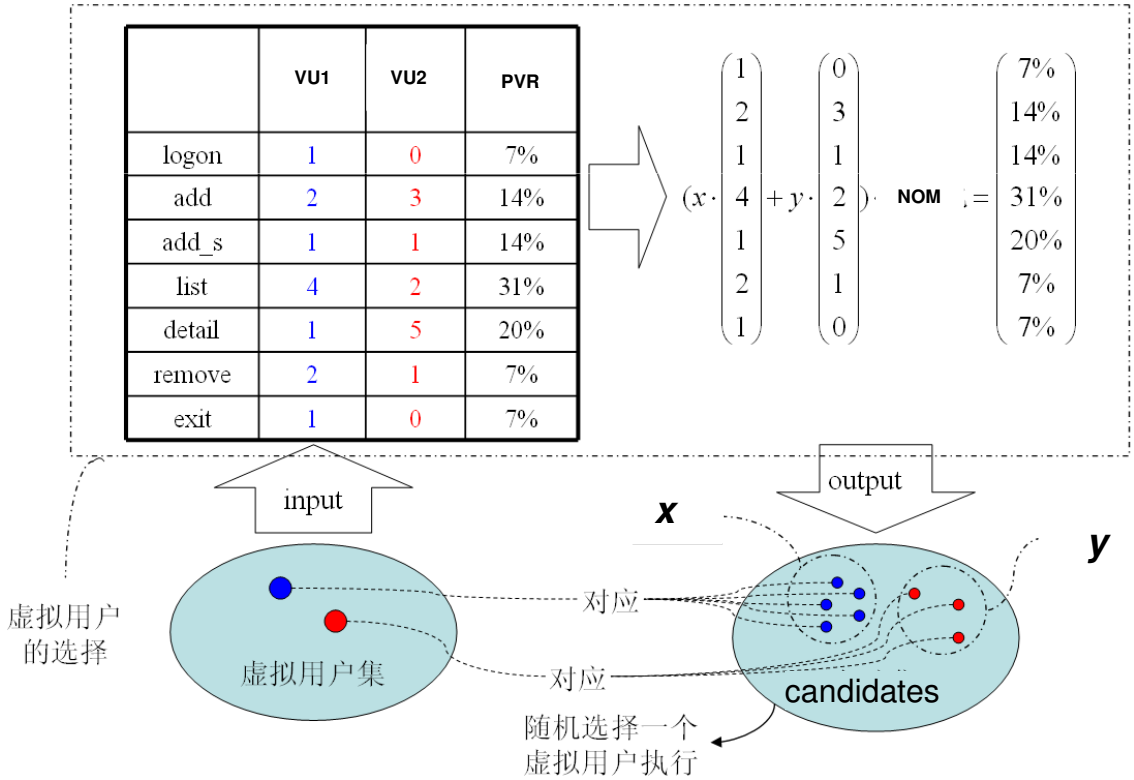
Load Profile Control Points

- Five control points in two stages of control
  - Static control (pre-testing): proportions of various VUs ➔ controlling session length, requests distri., byte distri.
  - Dynamic control (within-testing): #concurrent users, session interval, overall request interval, thinking time



Static Load Control

- Each kind of VU has pre-determined sequence of requests
- Given the proportions of various requests(PVR), to setup the number of different kinds of VUs

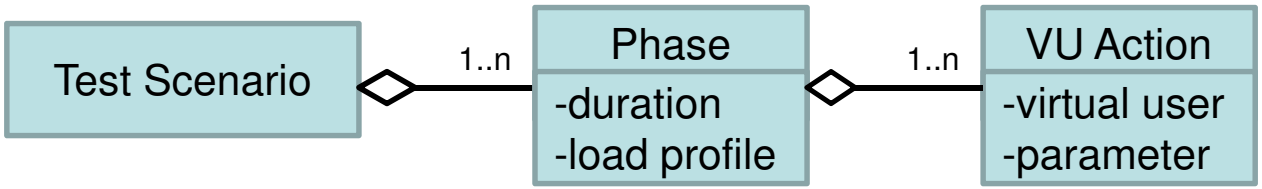


Dynamic Load Control

- We design seven ways to control the load dynamically.

Mode	Points to Control
1	Thinking time (distribution)
	#concurrent users (value)
2	Thinking time (distribution)
	Overall request interval (value)
3	Thinking time (distribution)
	Session interval (distribution)
4	Overall request interval (distribution)
5	Thinking time (distribution)
6	#concurrent users (value)
7	Session interval (distribution)

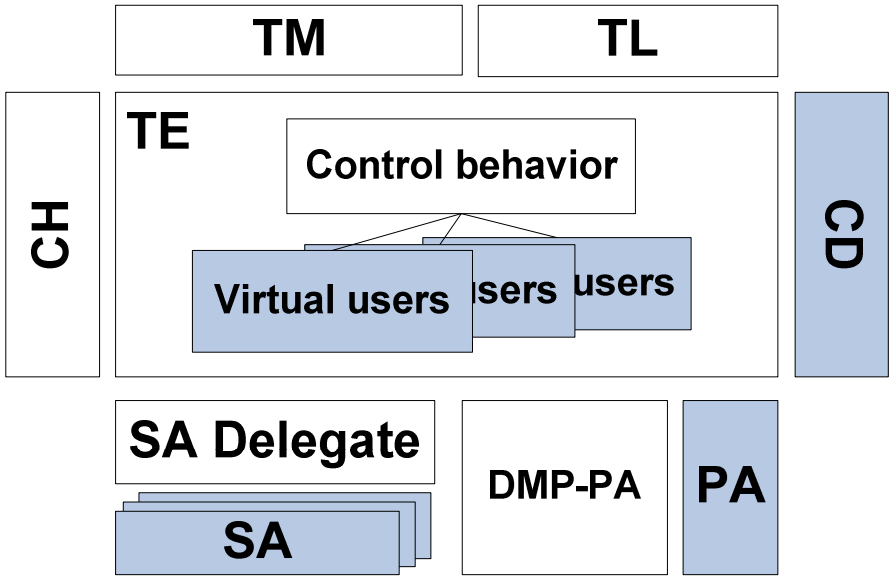
Performance Testing Scenario



- Provides phase-based load profile and control chance
- Provides control on the types of virtual users
- It can construct complex test scenario with required load profile, and simulate the real usages.

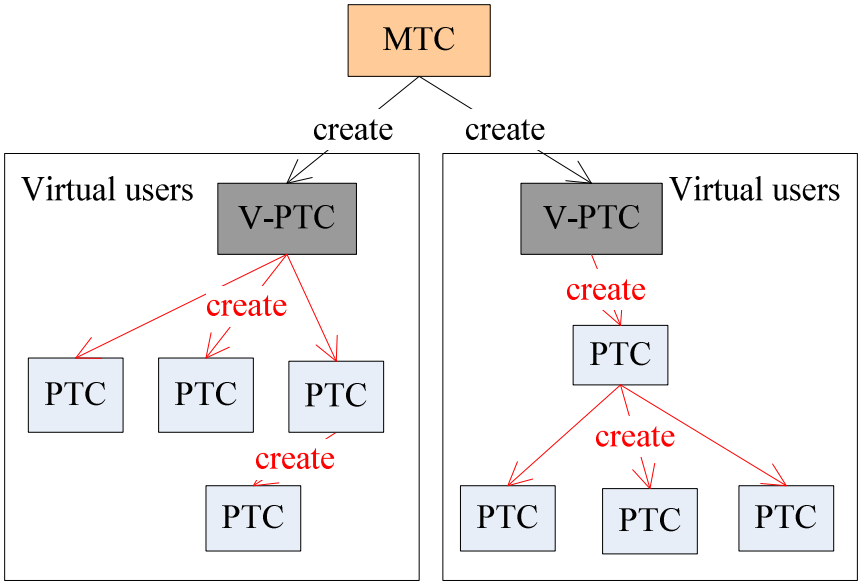
Test System Framework

- SA Delegate: Manage the communication connections to the SUT
- DMP-PA: Manages the Dynamic Module Parameters (DMP)
- TL: Manages virtual users, Monitors the test running, measures performance

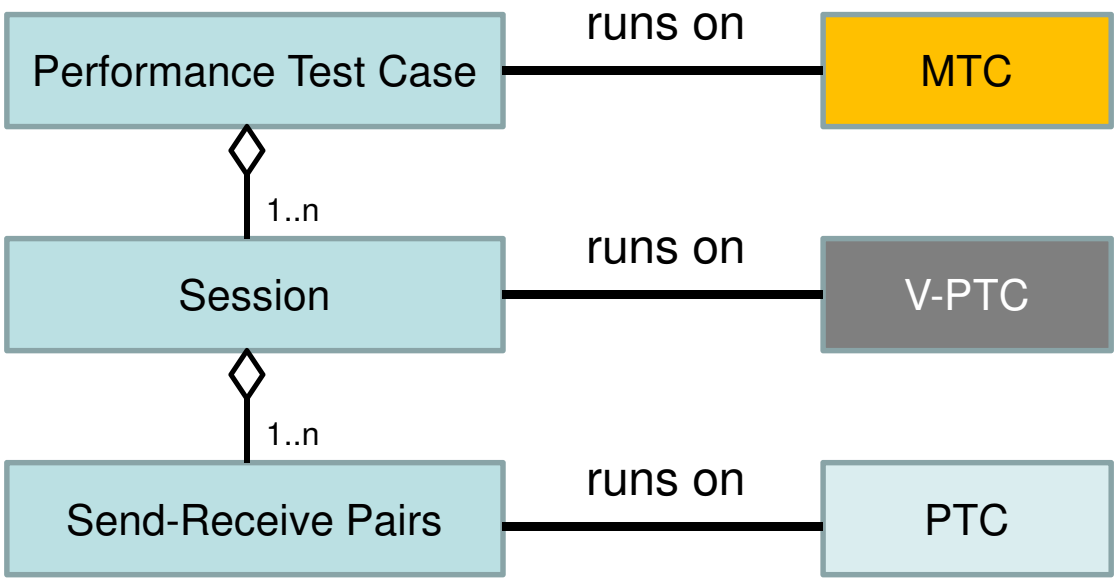


Virtual User Implementation

- Two levels of concurrency
  - Concurrency among virtual users
  - Concurrency within a virtual user
- Three levels of test components
  - MTC: manages the concurrency of virtual users
  - V-PTC: manages the concurrency within a virtual user, maintains a unique sessionID
  - PTC: manages the communication with other test component, maintains a unique componentID



Virtual User Implementation

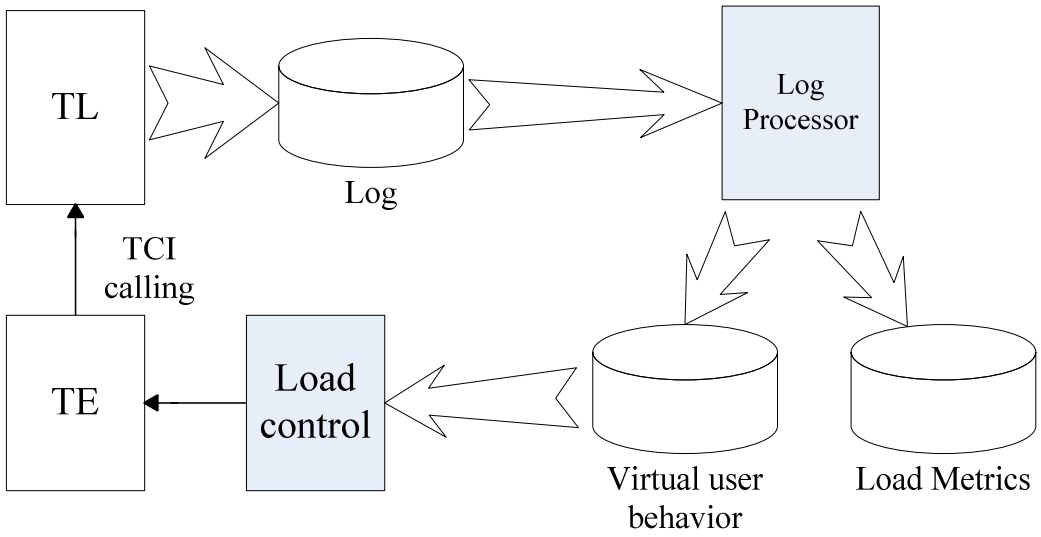


Load Control Implementation

- Load control is part of test system behavior.
  - A top level test case, keeps on running until test finished
- It monitors the relative ports to receive load control instruction
  - Read load parameters from DMP-PA
  - Create/Stop V-PTC
  - Start ready V-PTC with associated session
  - Control the V-PTC start interval
  - Control the number of live V-PTC

Monitoring of Virtual User Behavior

- Record the time to send and receive messages within a session
- Save as log file
- Calculate the load metrics from the log
- As feedback for adaptive load control



Performance Test Case

- Similarities with functionality test case
  - Similar test behavior (send/receive)
  - Use the same codec and adapter
- Differences from functionality test case
  - More powerful test configuration (three levels of test components)
  - Add the session hierarchy (runs on V-PTC)
  - Need dynamic module parameters management
- Automatically rewrite the test behavior in functionality test case

Conversion Rules

- Downgrade functionality test case

  testcase <*tc_name*>() *runs on compType_name1 {[statements]}* ➔

  *function subst_*<*tc_name*>(***charstring sessionID***) *runs on compType_name1{[statements]}*

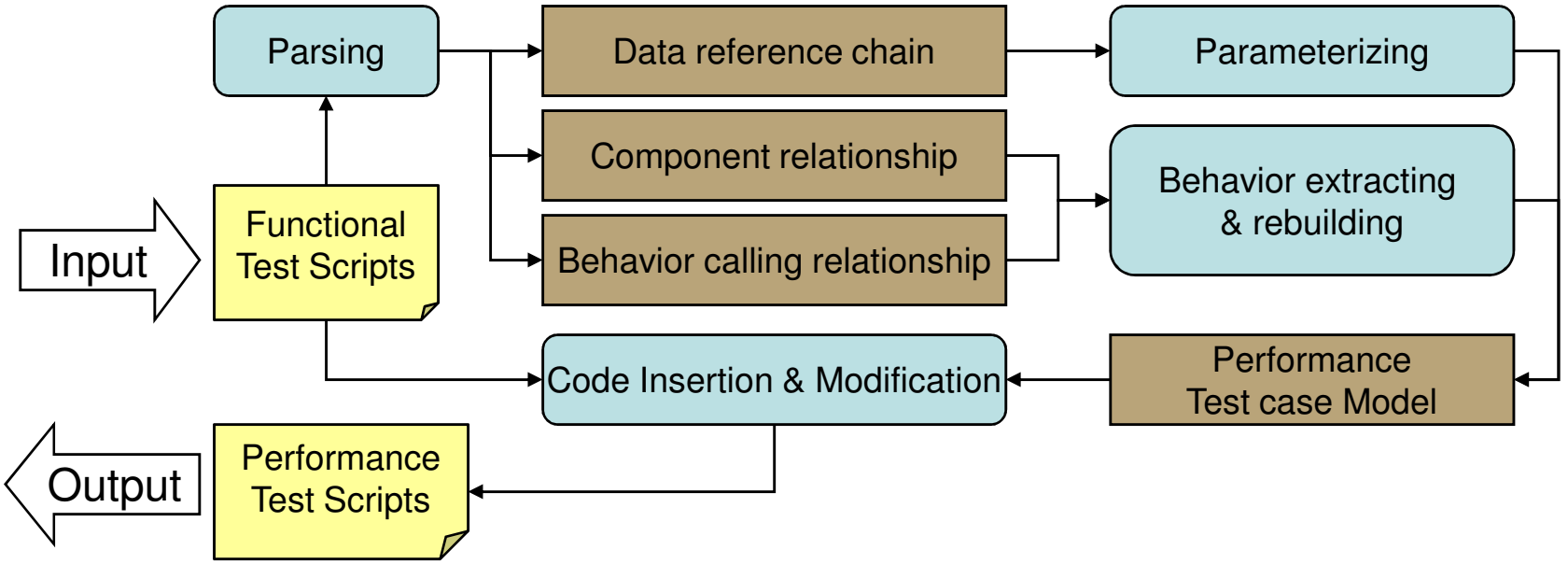
- Change references of *mtc* to *v-ptc*
- Get module parameter value from DMP-PA

  statementWith(ModuleParameter); ➔

  *var ModuleParaType mp_subst; mp_subst:=**getMpSubst**(sessionID); statementWith(ModuleParameter);*

  - ***getMpSubst*** *is an external function implemented in DMP-PA*

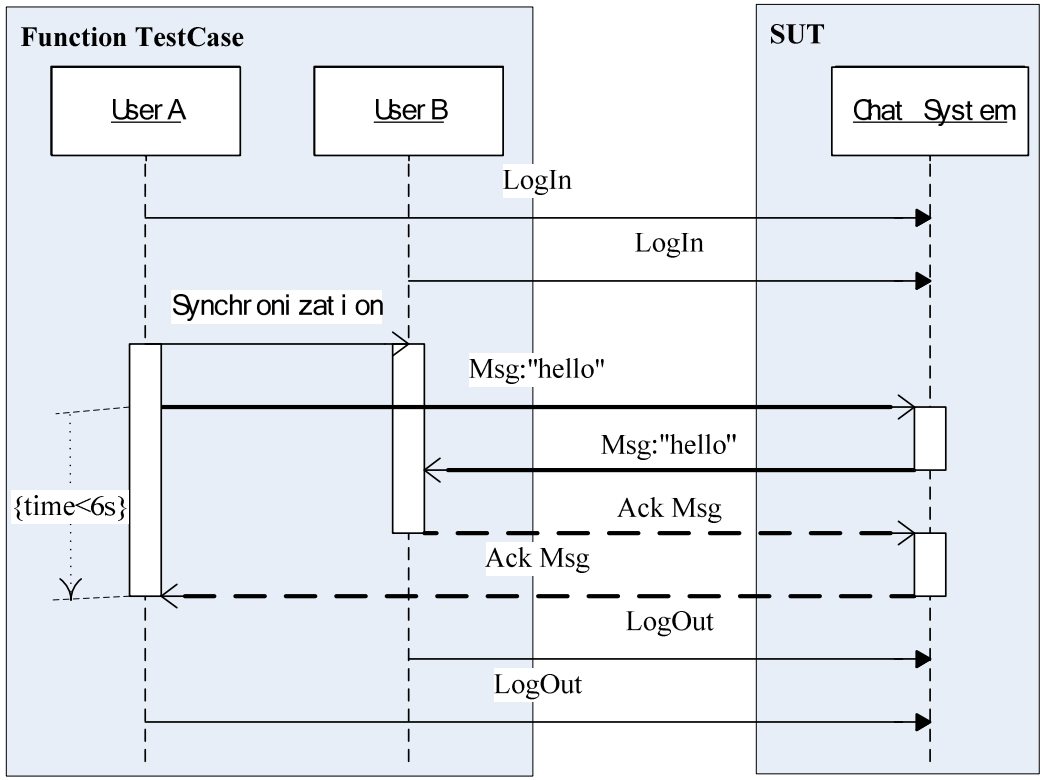
Conversion Implementation



*Parsing with TRex.

Experiment A

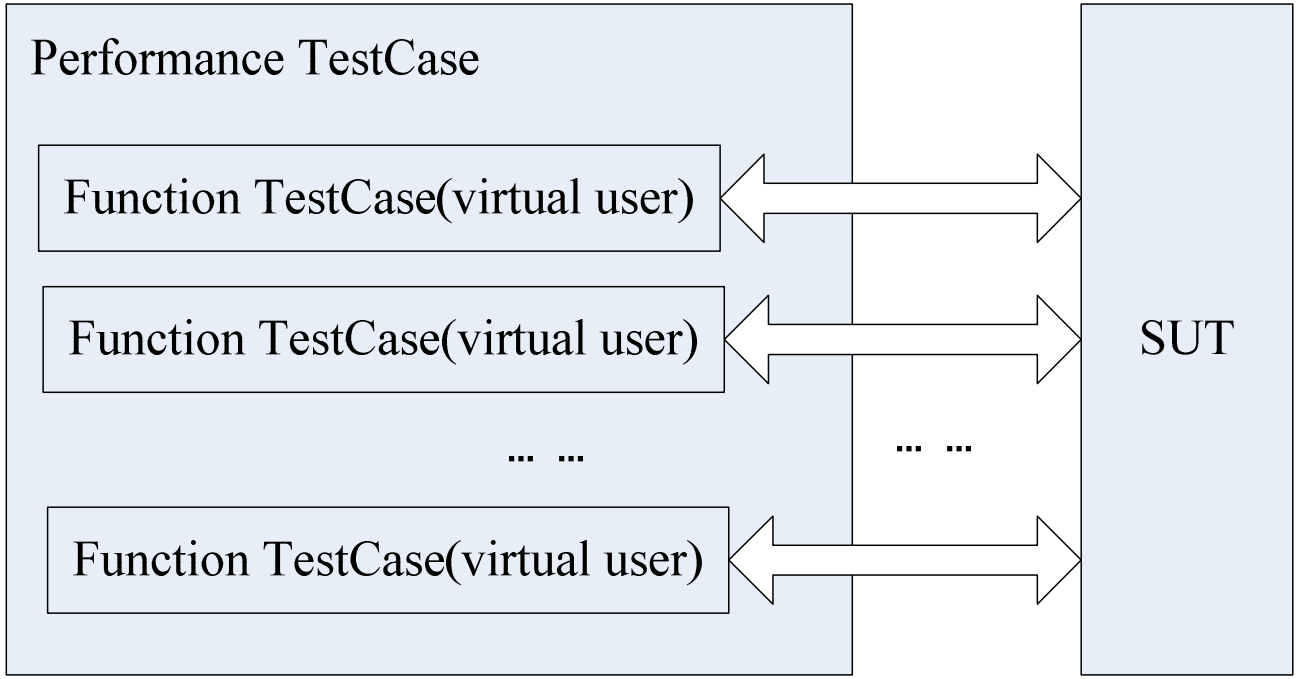
- Load testing for a simple chat system



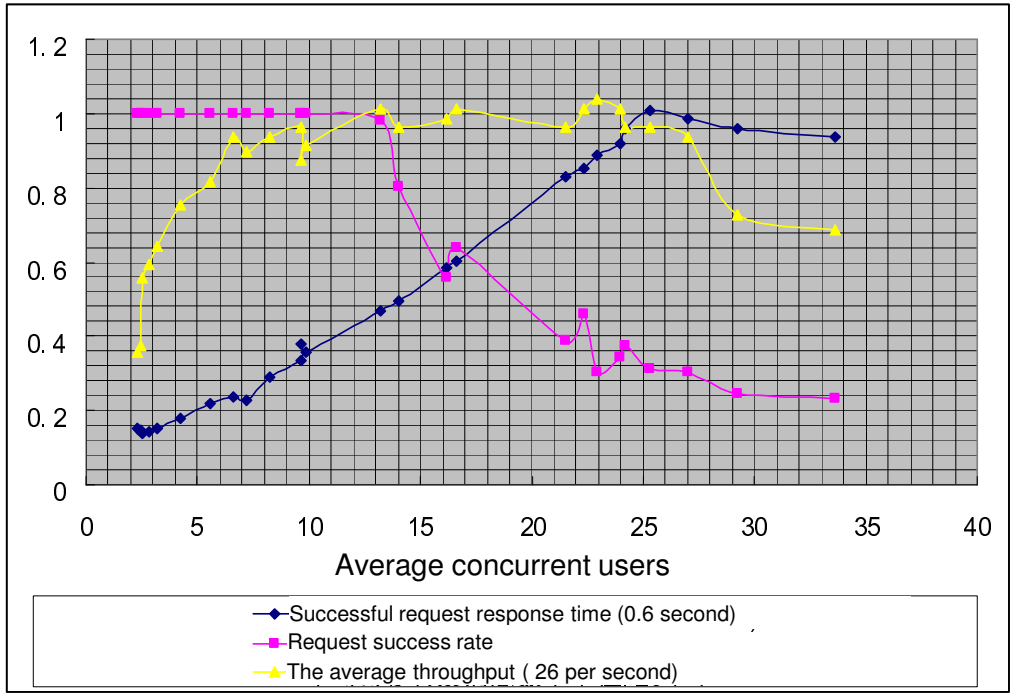
Functionality test case

Experiment A

- Load test case after automatic convert

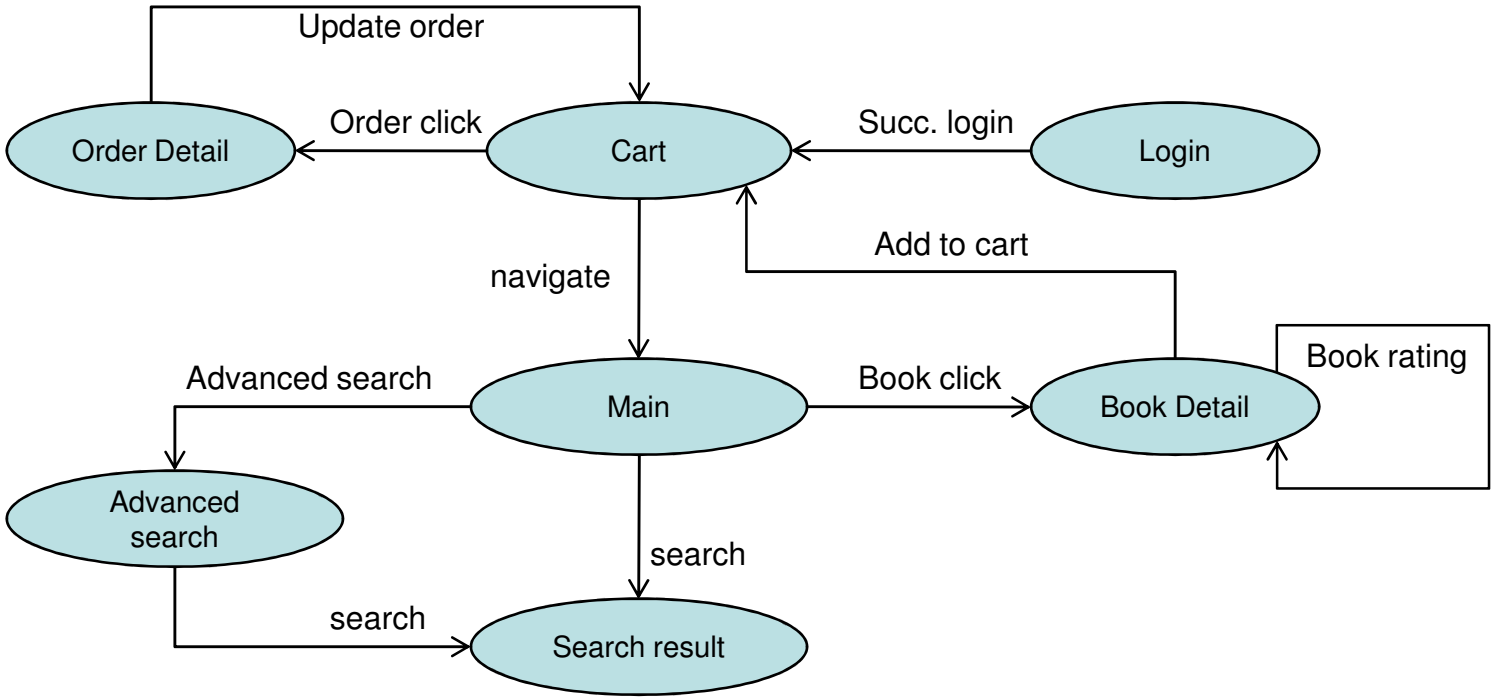


Experiment B



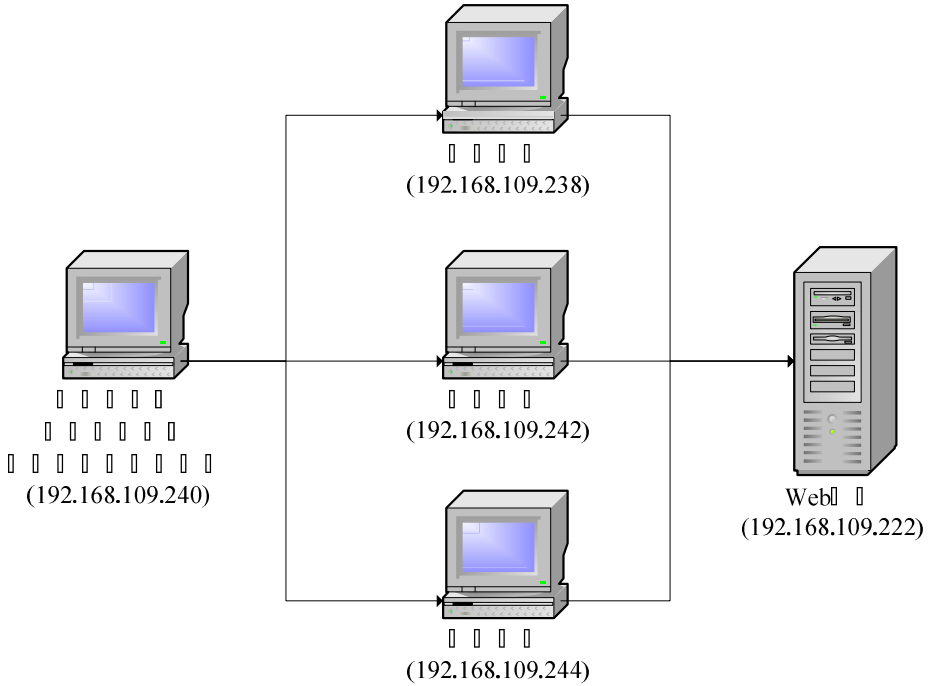
Experiment B

- SUT is a web-based bookstore
  - JSP for user interface and business logic
  - MySQL for data management and access



Experiment B

- 11 functionality test cases cover the flows
- Use our own distribution platform to support the test



Experiment B

XML

load

testcases

tc (11)

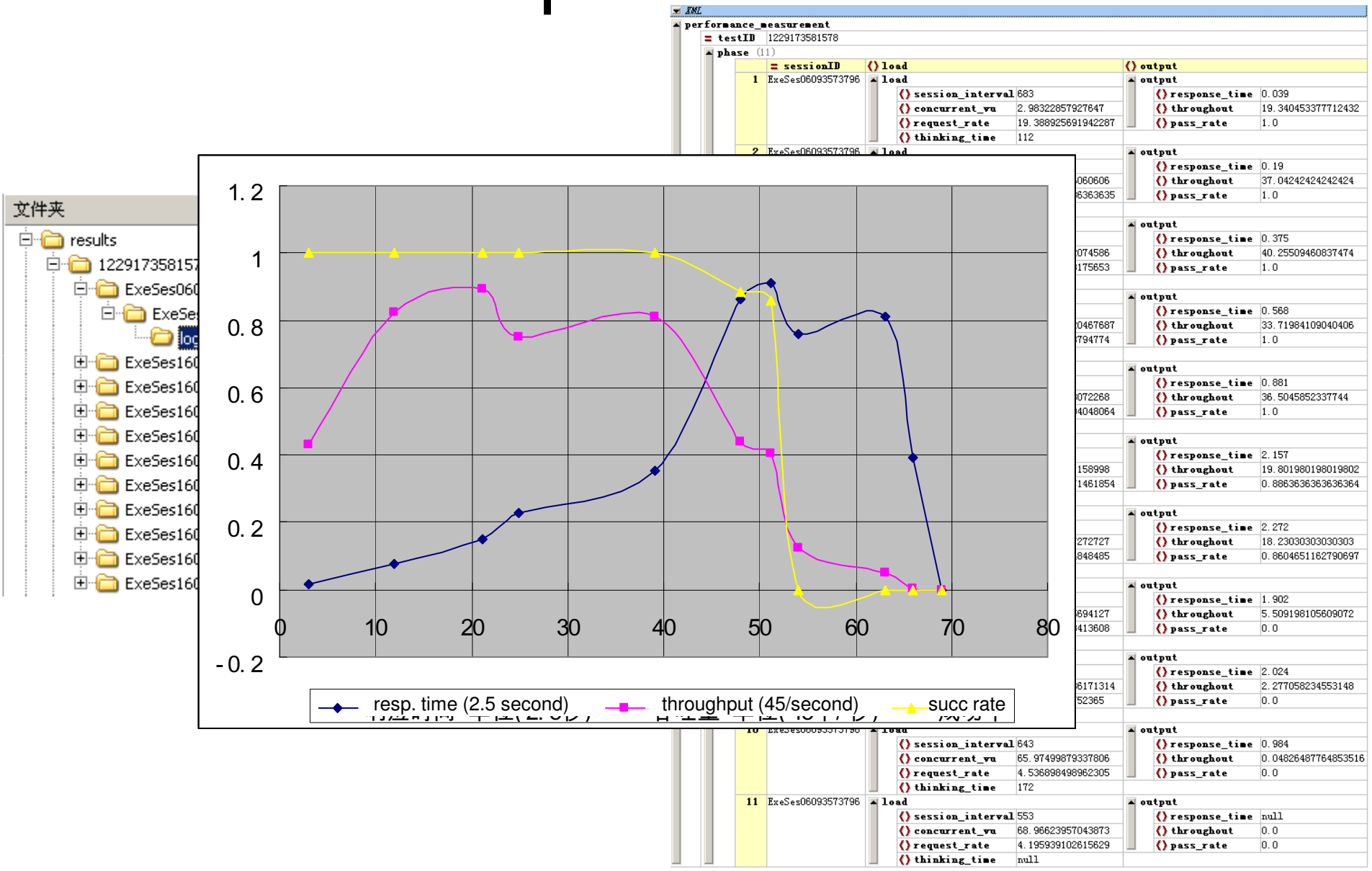
	name	len	mixWeight
1	tc001	15	1
2	tc002	25	1
3	tc003	11	1
4	tc004	15	1
5	tc005	19	1
6	tc006	12	1
7	tc007	13	1
8	tc_nologin_003	7	1
9	tc_nologin_004	11	1
10	tc_nologin_005	15	1
11	tc_nologin_006	8	1

campaign

phase (11)

	dur	sessionRate	concurrentVU	globalReqRate	thinkingTime
1	60	sessionRate name: null	3	globalReqRate name: null	thinkingTime name: average para: 0.1
2	60	sessionRate name: null	12	globalReqRate name: null	thinkingTime name: average para: 0.1
3	60	sessionRate name: null	21	globalReqRate name: null	thinkingTime name: average para: 0.1
4	60	sessionRate name: null	30	globalReqRate name: null	thinkingTime name: average para: 0.1
5	60	sessionRate name: null	39	globalReqRate name: null	thinkingTime name: average para: 0.1
6	60	sessionRate name: null	48	globalReqRate name: null	thinkingTime name: average para: 0.1
7	60	sessionRate name: null	51	globalReqRate name: null	thinkingTime name: average para: 0.1
8	60	sessionRate name: null	54	globalReqRate name: null	thinkingTime name: average para: 0.1
9	60	sessionRate name: null	63	globalReqRate name: null	thinkingTime name: average para: 0.1
10	60	sessionRate name: null	66	globalReqRate name: null	thinkingTime name: average para: 0.1
11	60	sessionRate name: null	69	globalReqRate name: null	thinkingTime name: average para: 0.1

Experiment B



Conclusion

- This work is supported by 2 National High-Tech program projects.
- We develop the distributed load testing in TTCN-3
  - Profile based load generation
  - Adaptive load control
  - Reuse existing functionality test cases
- TTCN-3 language provides great flexibility and good structure for load testing.

Questions?

- Thanks for your attention!