

# 敏捷开发模式下复杂企业产品的质量保证与持续测试

Quality Assurance for complex enterprise product in agile model & Continuous Testing

陆明刚

2017.07.22

# About Me



- 易安信中国研发中心中端存储部，架构师
- 曾任职趋势科技中国研发中心，技术经理
- 《Java性能调优权威指南》译者之一
- 10+ 项中国，美国专利
- 爱好阅读，摄影，羽毛球，跑步

# Agenda



EMC MRES product portfolio



Challenges that we are facing



What's BBT/CCT



Continuous Test Execution Environment (CTEE)



Summary



Q & A

# EMC MRES Product Portfolio

## SC Series

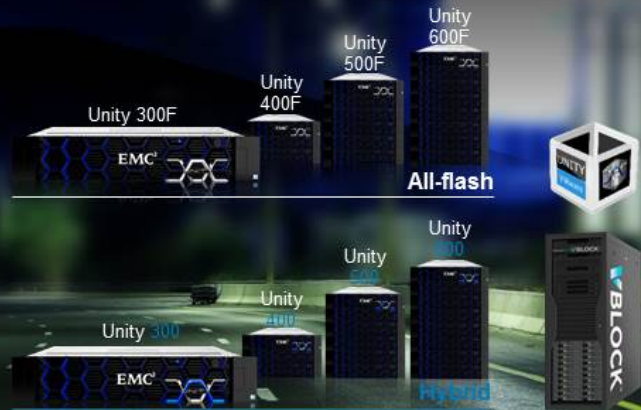
Value-optimized  
Mid-Market Proven



SC Series

## Unity

New Mid-Range  
Simple, Modern, Flexible, Affordable



EMC Unity

DELL EMC

# EMC MRES Product Portfolio (Cont.)

## Dell EMC Unity Success since May 2016 Launch



Bookings: \$1 Billion ▲



Capacity: 1,070 PB ▲



Systems: 11,500 ▲

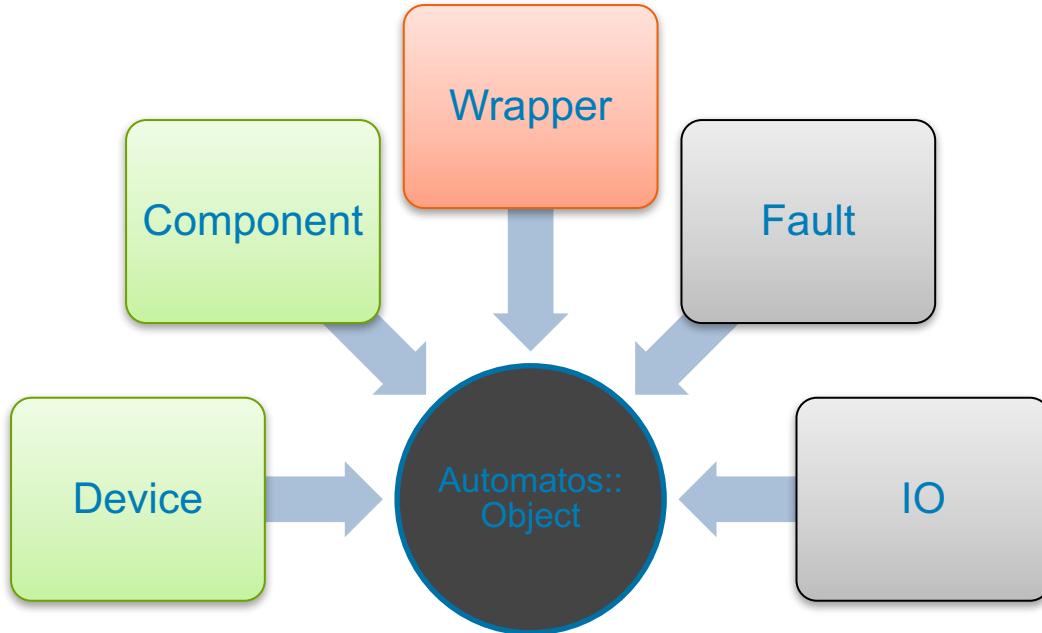


Customers: 5,800 ▲

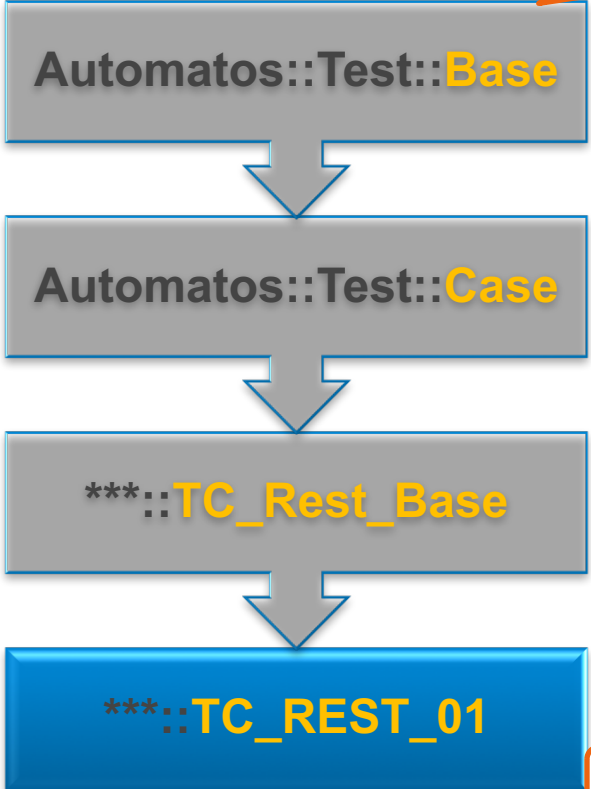
# Challenges that we are facing

- Release schedule tighten, faster deliver pressure
  - 1 Year → 3 Month
- Large number of legacy test cases and various automation framework and tools
  - 20000+ Test Cases
  - 5+ automation test framework/tools
    - ✓ Meta session
    - ✓ QTP
    - ✓ QES
    - ✓ ....
- Complex test scenarios
  - Array, Host, Application etc.

# Automatos Automation Framework



# Hierarchy



Common modules

- Subroutines
  - preTestCase
  - postTestCase

Interfaces

- Subroutines
  - createMetadata
  - preTestCase
  - postTestCase
  - logPropertyInfo

Get global parameters(TestSet)

Implementation.

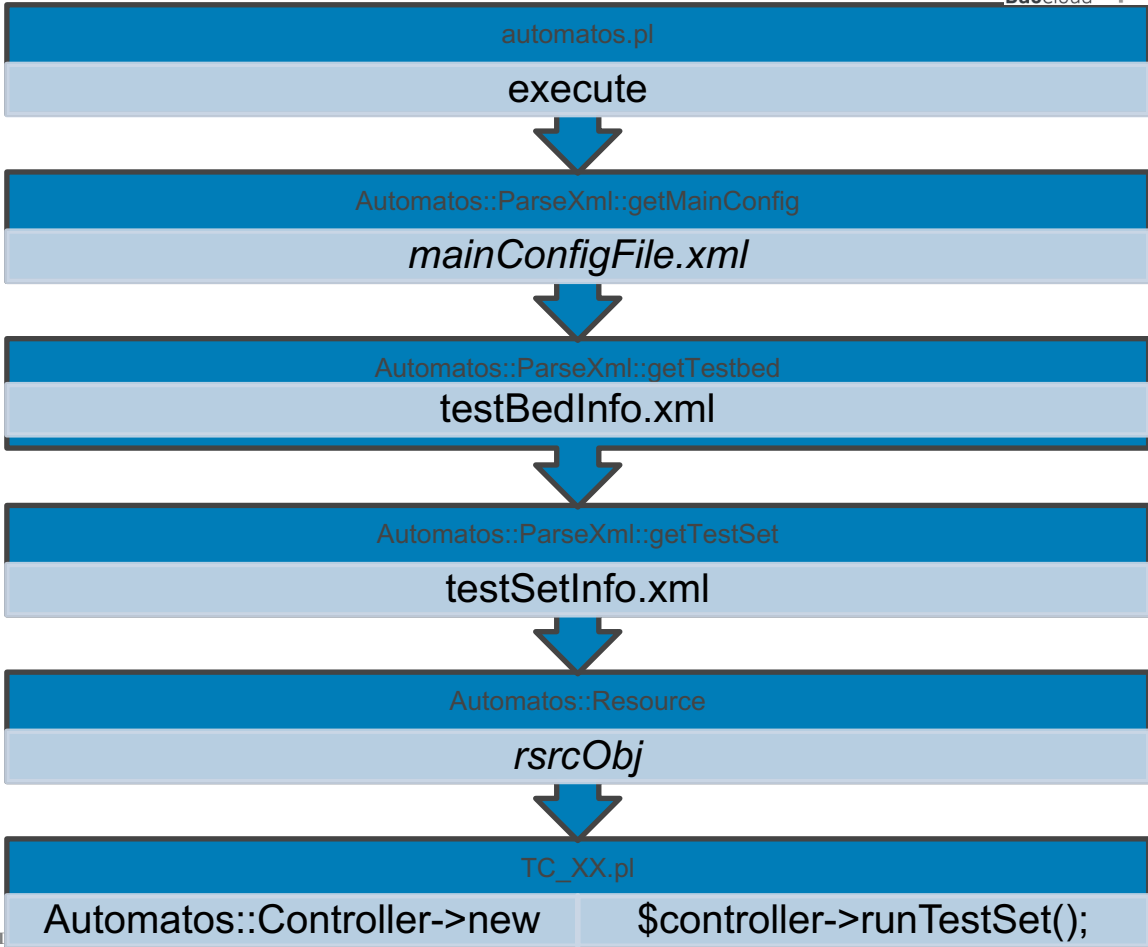
- Subroutines
  - createMetadata
  - main

Get local parameters(TestSet)

`$self->getParameter(name => 'service');`



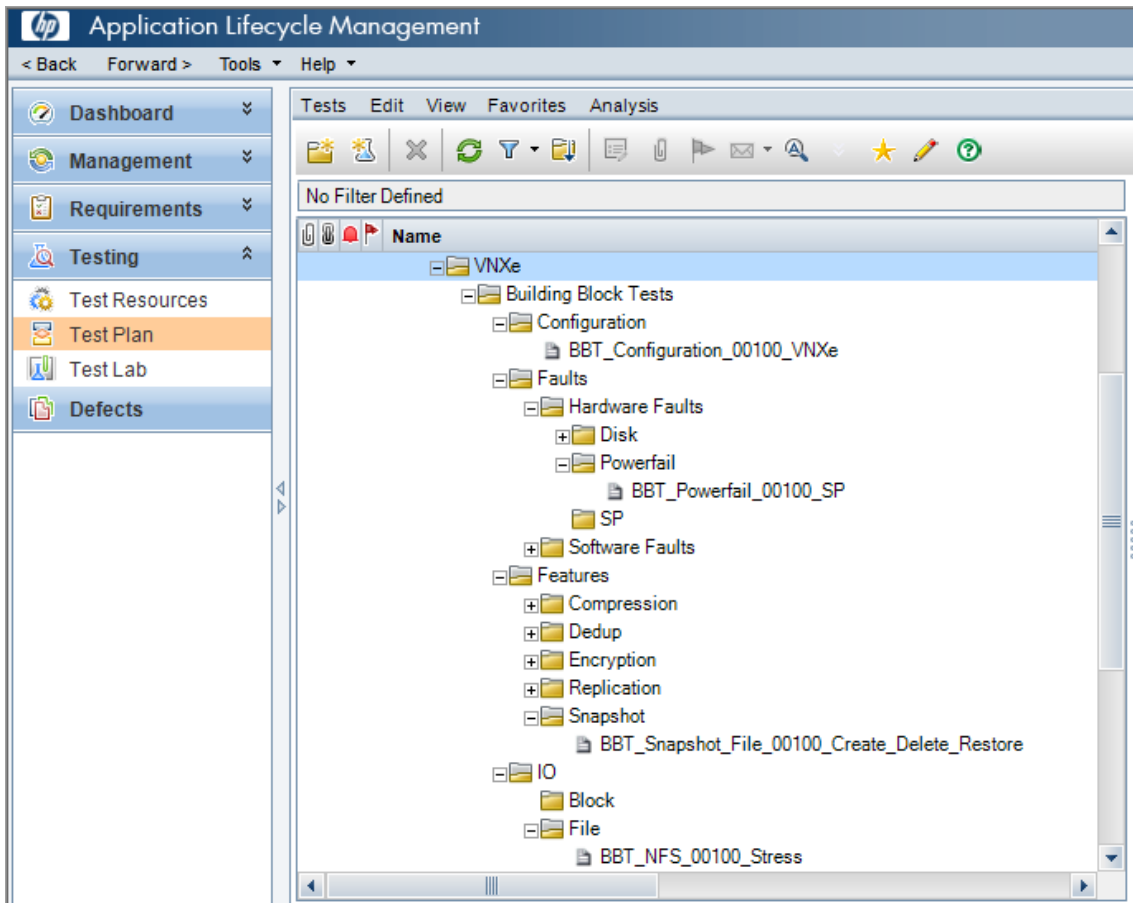
# Workflow



# BBT and CCT

- **Building Block Test Case/Module (BBT)**
  - Common modules for building complex Test Cases
  - Implemented as an Automatos Test Case
  - Assembled into Automatos Test Sets
  - Automatos support for parallel execution required
  
- **Complex Combination Test Case (CCT)**
  - Represents a collection of BBTs
  - CCT describes how BBTs are used
  - Automatos support for CCT reporting required

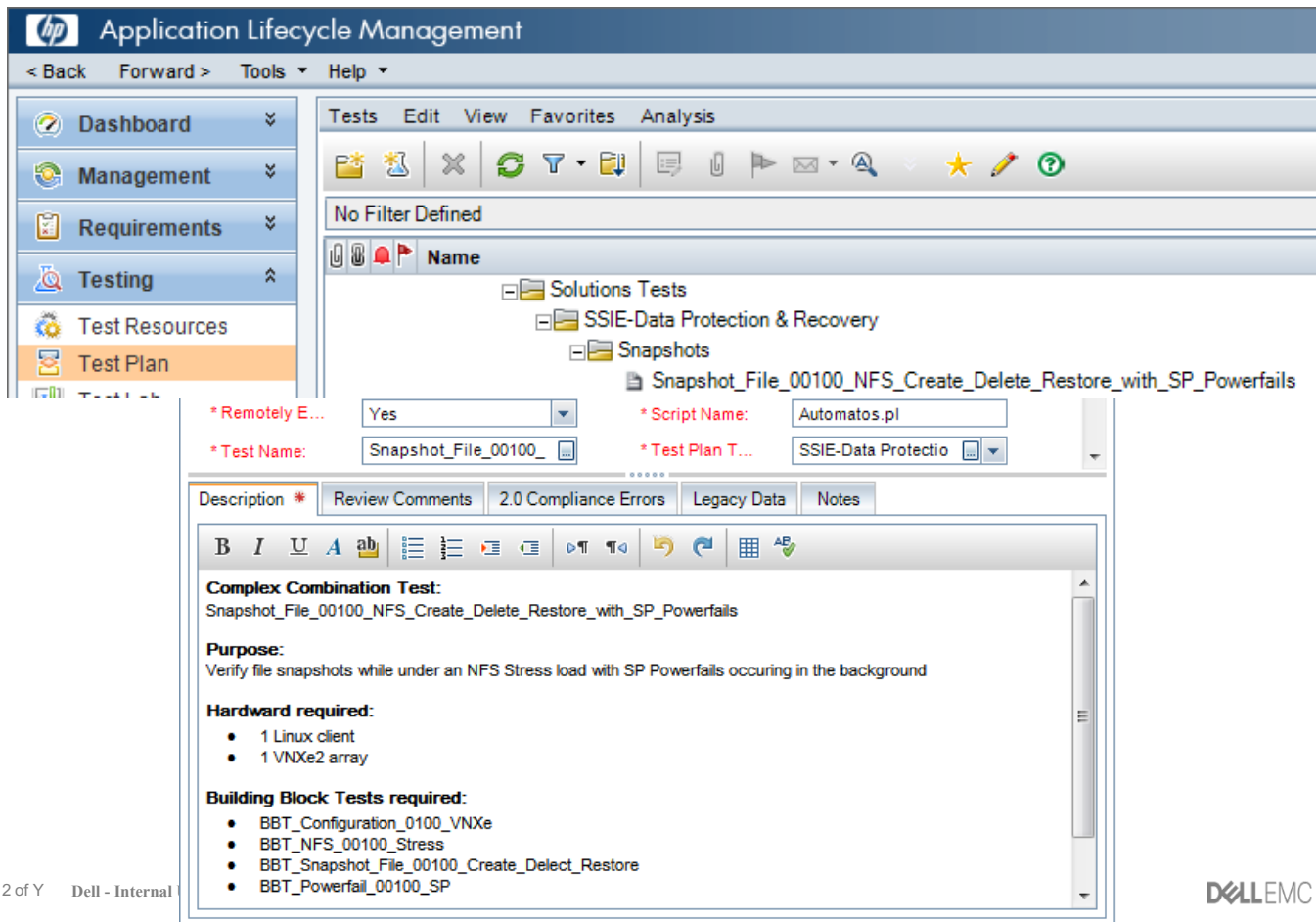
# BBT in Test Plan Example



The screenshot displays the HP Application Lifecycle Management (ALM) interface. The left sidebar shows a navigation menu with the following items: Dashboard, Management, Requirements, Testing, Test Resources, Test Plan (highlighted), Test Lab, and Defects. The main window title is "Application Lifecycle Management" and includes navigation buttons for Back, Forward, Tools, and Help. Below the title bar, there are tabs for Tests, Edit, View, Favorites, and Analysis. A toolbar contains various icons for actions like adding, deleting, refreshing, filtering, and saving. The main content area shows a tree view of test items under the heading "No Filter Defined". The tree structure is as follows:

- VNXe
  - Building Block Tests
    - Configuration
      - BBT\_Configuration\_00100\_VNXe
    - Faults
      - Hardware Faults
        - Disk
        - Powerfail
          - BBT\_Powerfail\_00100\_SP
        - SP
      - Software Faults
    - Features
      - Compression
      - Dedup
      - Encryption
      - Replication
      - Snapshot
        - BBT\_Snapshot\_File\_00100\_Create\_Delete\_Restore
    - IO
      - Block
      - File
        - BBT\_NFS\_00100\_Stress

# CCT in Test Plan Example



The screenshot displays the HP Application Lifecycle Management (ALM) interface. The left sidebar shows navigation options: Dashboard, Management, Requirements, Testing, Test Resources, and Test Plan. The main area shows a test plan configuration for a test named "Snapshot\_File\_00100\_NFS\_Create\_Delete\_Restore\_with\_SP\_Powerfails".

**Test Configuration:**

- \* Remotely Execute: Yes
- \* Script Name: Automatos.pl
- \* Test Name: Snapshot\_File\_00100\_NFS\_Create\_Delete\_Restore\_with\_SP\_Powerfails
- \* Test Plan Name: SSIE-Data Protection & Recovery

**Description:** Review Comments, 2.0 Compliance Errors, Legacy Data, Notes

**Complex Combination Test:**  
Snapshot\_File\_00100\_NFS\_Create\_Delete\_Restore\_with\_SP\_Powerfails

**Purpose:**  
Verify file snapshots while under an NFS Stress load with SP Powerfails occurring in the background

**Hardware required:**

- 1 Linux client
- 1 VNXe2 array

**Building Block Tests required:**

- BBT\_Configuration\_0100\_VNXe
- BBT\_NFS\_00100\_Stress
- BBT\_Snapshot\_File\_00100\_Create\_Delect\_Restore
- BBT\_Powerfail\_00100\_SP

# CCT/BBT in Test Lab Testset Example



hp Application Lifecycle Management

< Back Forward > Tools Help

Dashboard Management Requirements Testing Test Resources Test Plan Test Lab Defects

Test Sets Test Runs

Test Sets Edit View Tests Favorites Analysis

No Filter Defined

- Kittyhawk\_Plus
  - CFT-Duration
  - CIT
  - Feature Testing
  - Mass Upgrades
  - B to C
    - Data Protection & Recovery
      - Static
        - Snapshot File

Select Tests Run Run Test Set

Details Execution Grid Execution Flow Automation Attachments Linked Defects History

Test: Test Name		Status
Snapshot_File_00100_NFS_Create_Delete_Restore_with_SP_Powerfails	No Run	
BBT_Configuration_00100_VNXe	No Run	
BBT_NFS_00100_Stress	No Run	
BBT_Snapshot_File_00100_Create_Delete_Restore	No Run	
BBT_Powerfail_00100_SP	No Run	

# CTE<sup>2</sup> Roadmap



Q1  
2016

## CTE<sup>2</sup> 1.0

- Centralized, Fault Tolerant Jenkins Server
- Test Beds (Arrays, Hosts, Appliances, etc.) defined in Jenkins Server
- Hardcoded Test Sets ↔ Test Beds mappings – no resource pooling/management. (i.e. No CACTUS)
- Manual configuration, setup, recovery and MBU
- Test Results automatically stored in UTMS such that RADAR can automatically report results
- Automatic execution of all cycle tests 1x
- Automatic continuous execution of all cycle tests (over and over ...)
- Automatic updating of test set IDs in XML files. (Cycle to Cycle transitions)
- Simple Orchestration (Jenkins, other?) – Orchestration 0.1

Q2  
201X

## CTE<sup>2</sup> 1.2

- CACTUS Full management of test beds (limited resource optimizations)
- Automatic configuration and setup of remaining test beds.
- CACTUS/Centralized Health Service Integrated
- Manual Recovery of test beds
- Orchestration 1.0

QX  
201X

## CTE<sup>2</sup> 1.3

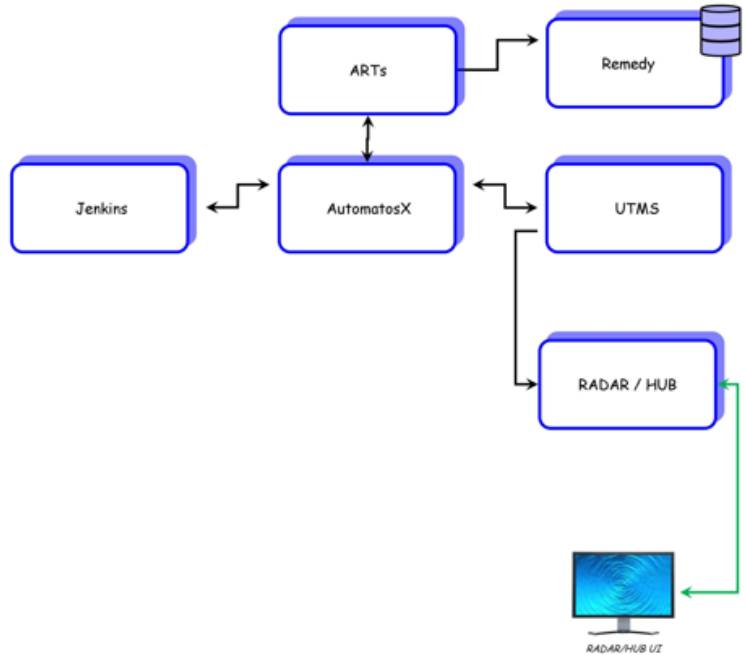
- Automated Recovery of test beds

## CTE<sup>2</sup> 1.1

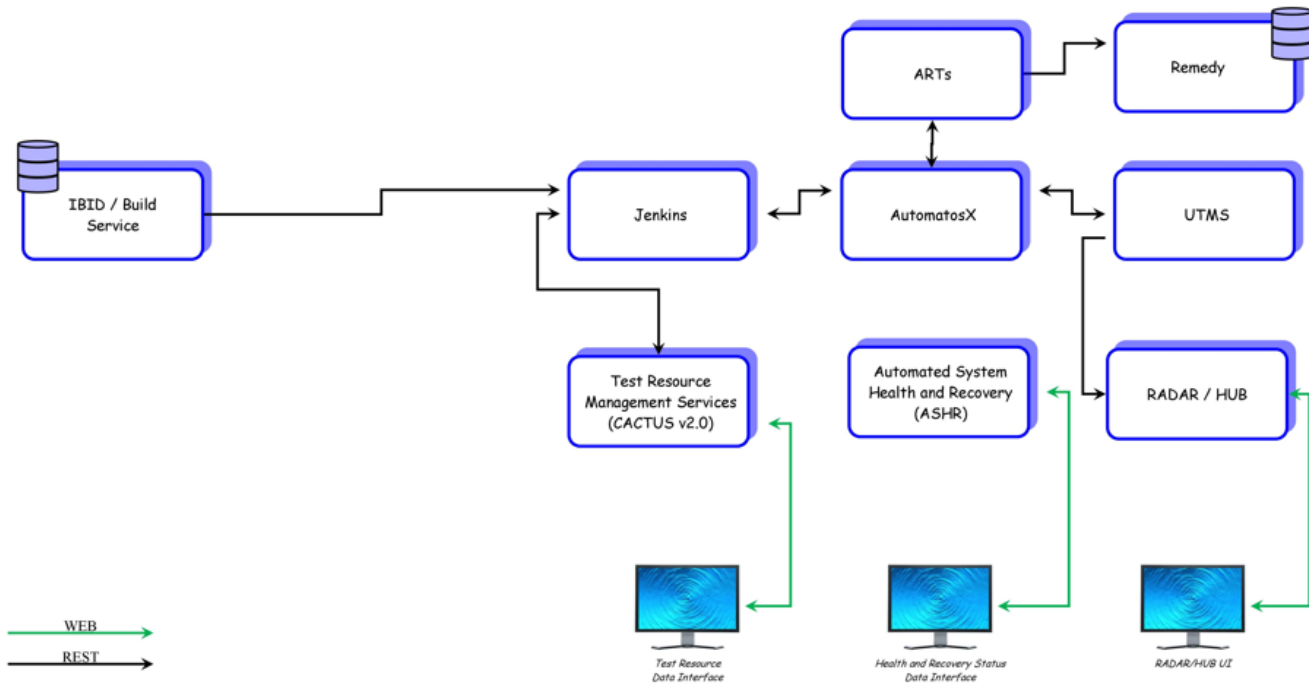
- Automatic MBU of arrays
- Manual Recovery of test beds
- Standalone Centralized Health Service
- Test Beds (Arrays, Hosts, Appliances, etc.) defined in Centralized Health Service
- CACTUS w/limited management of test beds (no resource optimizations)



# CTE<sup>2</sup>—Continuous Test Execution Environment—V1.0

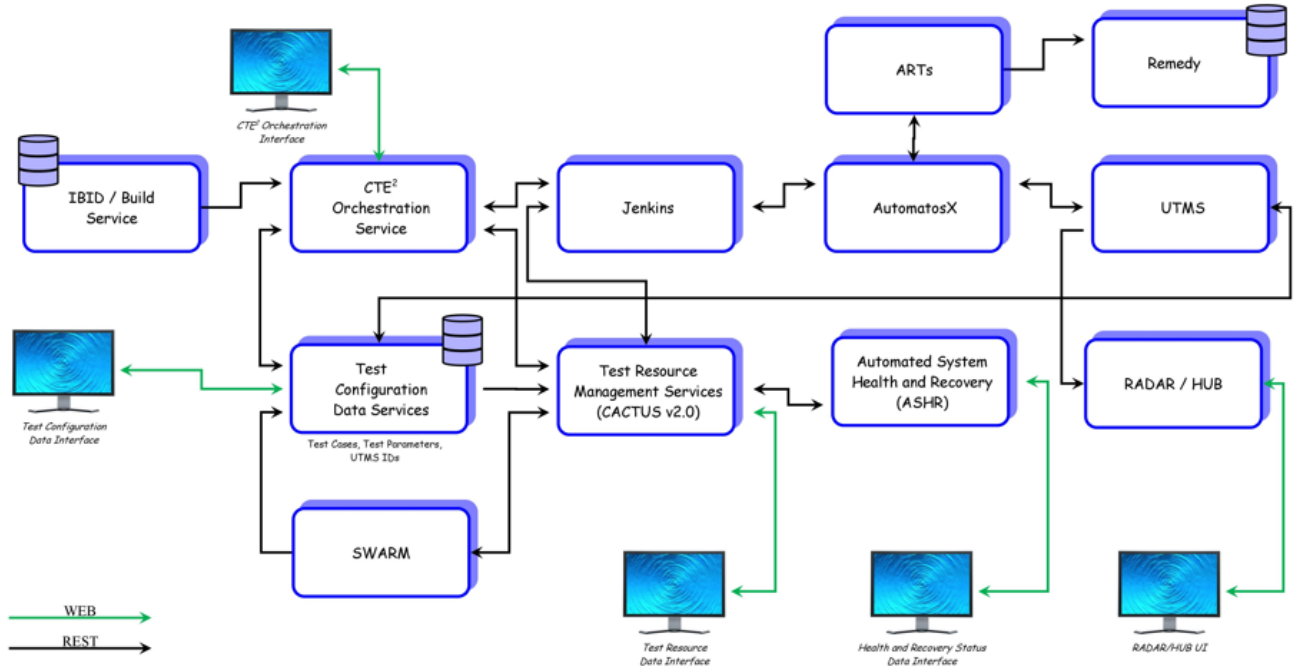


# CTE<sup>2</sup>—Continuous Test Execution Environment—V1.1





# CTE<sup>2</sup>—Continuous Test Execution Environment—V1.2





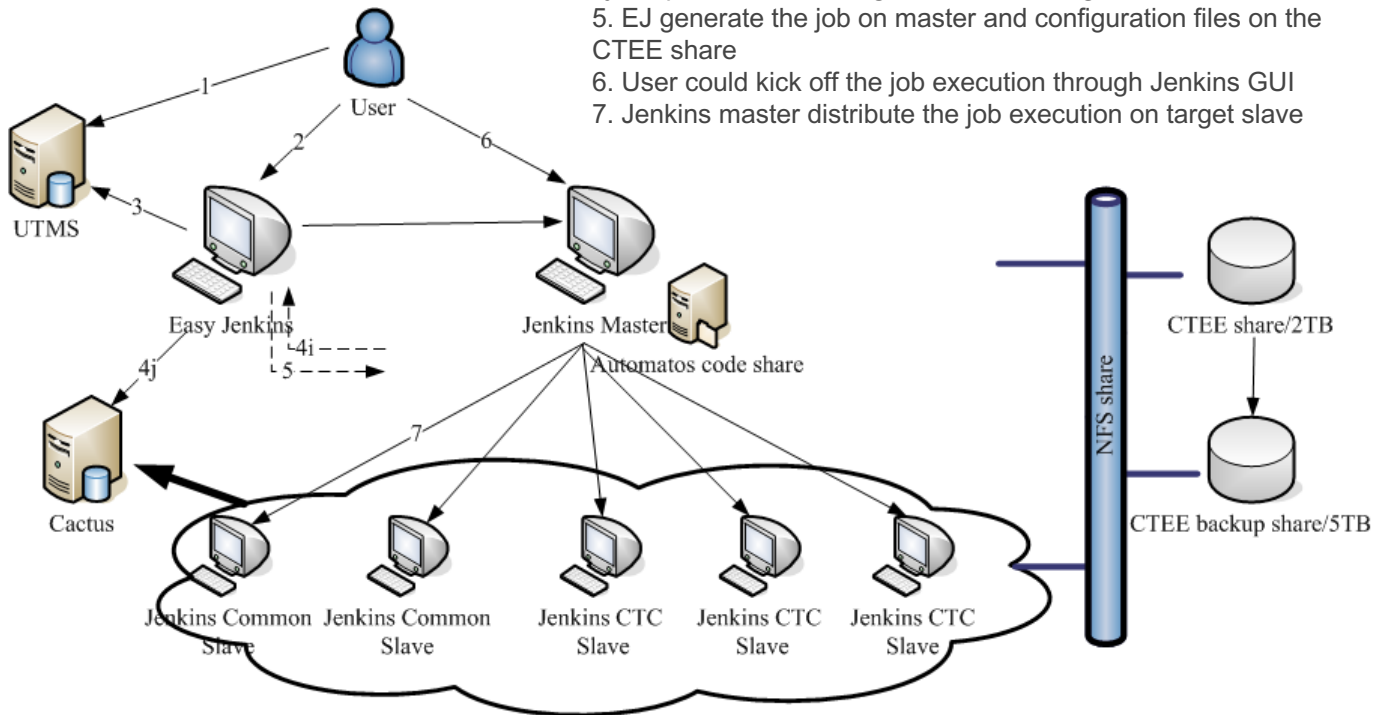
DaoCloud



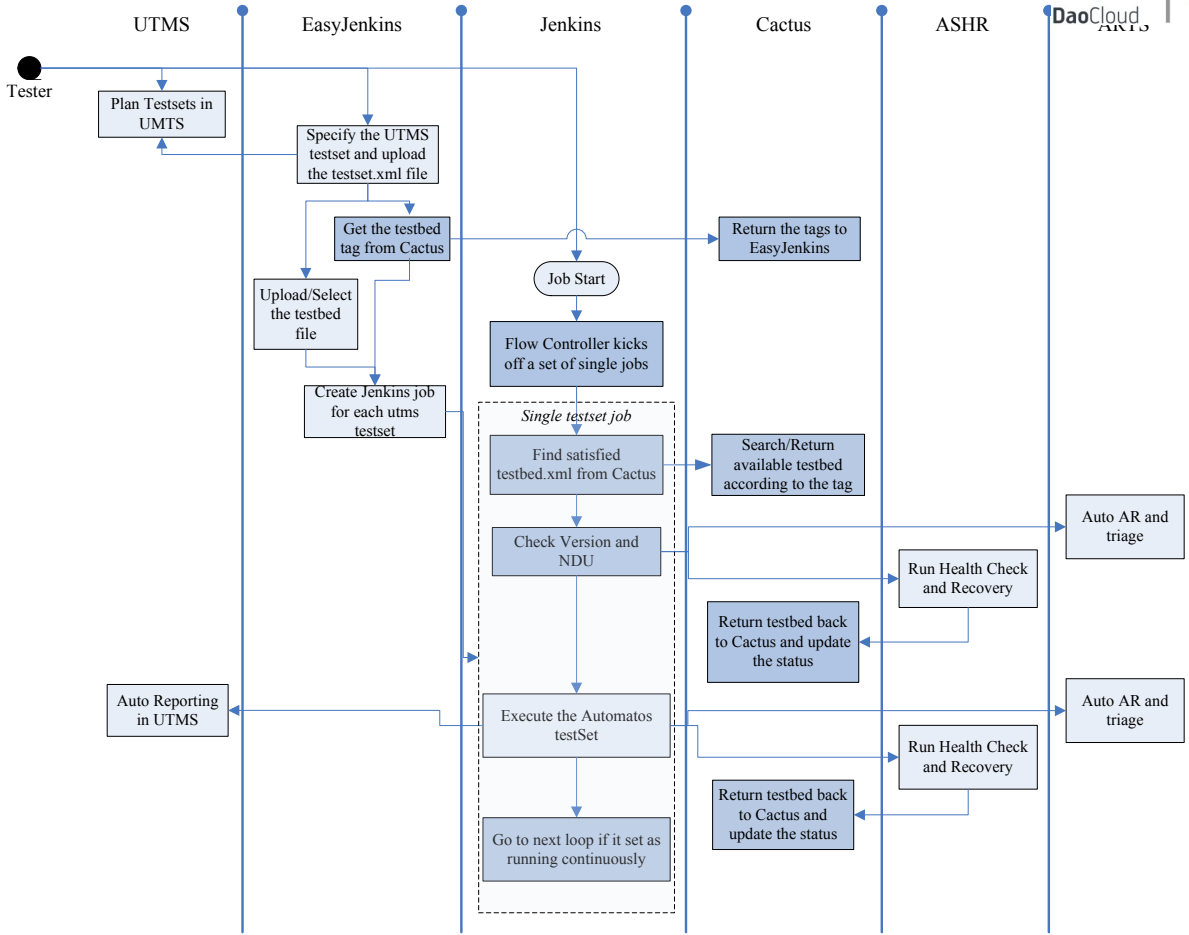
知识分享平台

# CTE<sup>2</sup> Overall Topology

1. User create testsets in the test lab
2. User create Jenkins jobs for each testset via EasyJenkins(EJ)
3. EJ would get the testset info from the UTMS
- 4i. EJ get testset/bed files from user or let user select existing files
- 4j. EJ provide Cactus tag for user selecting
5. EJ generate the job on master and configuration files on the CTEE share
6. User could kick off the job execution through Jenkins GUI
7. Jenkins master distribute the job execution on target slave



# CTE<sup>2</sup> Overall Flow Chart




EMC<sup>2</sup> Continuous Test Execution Environment

Home Testbed Jenkins Job Array Host AR Show summary in 2016-15

Program Falcon

## MRQE CTE<sup>2</sup> DASHBOARD

CTEE Best Practice



# MIDRANGE QUALITY ENGINEERING

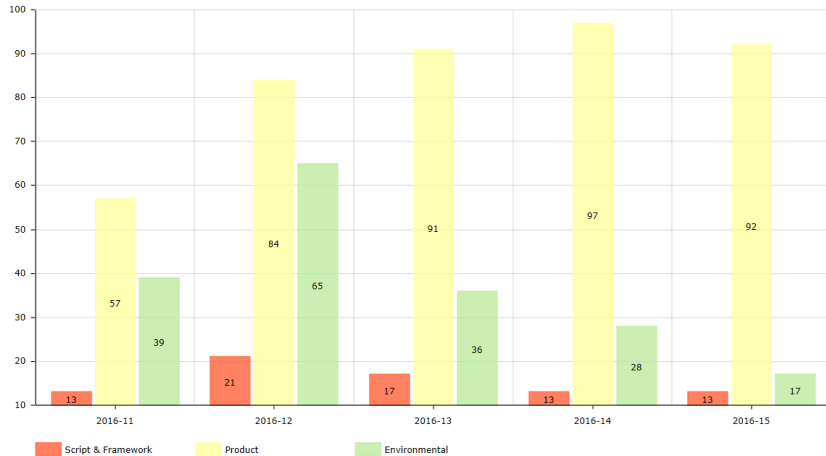
*YOU DEFINE*

Jenkins Slaves	73	Testbeds	339	Jenkins Jobs	271
CTCs	12	Arrays	76	Hosts	415



# CTEE Jenkins Job Summary

CTC Name	Jenkins Jobs	Defined Test Cases (UTMS)	Defined Test Cases (Jenkins)	Passed TC of Last Run	Executed TC	Pass Rate	Non-Compliant Jobs
MRQE_APPDB	19	28	10	10	23	35.71%	0
MRQE_DM	19	131	98	63	119	48.09%	0
MRQE_DPR	0	1	0	0	0	0.00%	0
MRQE_Durability	6	16	14	10	38	62.50%	0
MRQE_Endurance	37	6	63	1	114	16.67%	0
MRQE_Install_and_Config	8	26	3	0	4	0.00%	0
MRQE_INTEROP_EMC	9	21	6	2	5	9.52%	0
MRQE_INTEROP_OS	6	25	5	3	4	12.00%	0
MRQE_LargeScale	9	35	4	0	8	0.00%	0
MRQE_Platform	75	205	52	26	90	12.68%	0
MRQE_Serviceability	61	177	120	102	226	57.63%	0
MRQE_Stress	9	93	52	14	53	15.05%	0
MRQE_VC	13	82	12	9	17	10.98%	4
<b>Grand Total</b>	<b>271</b>	<b>846</b>	<b>439</b>	<b>240</b>	<b>701</b>	<b>28.37%</b>	<b>4</b>

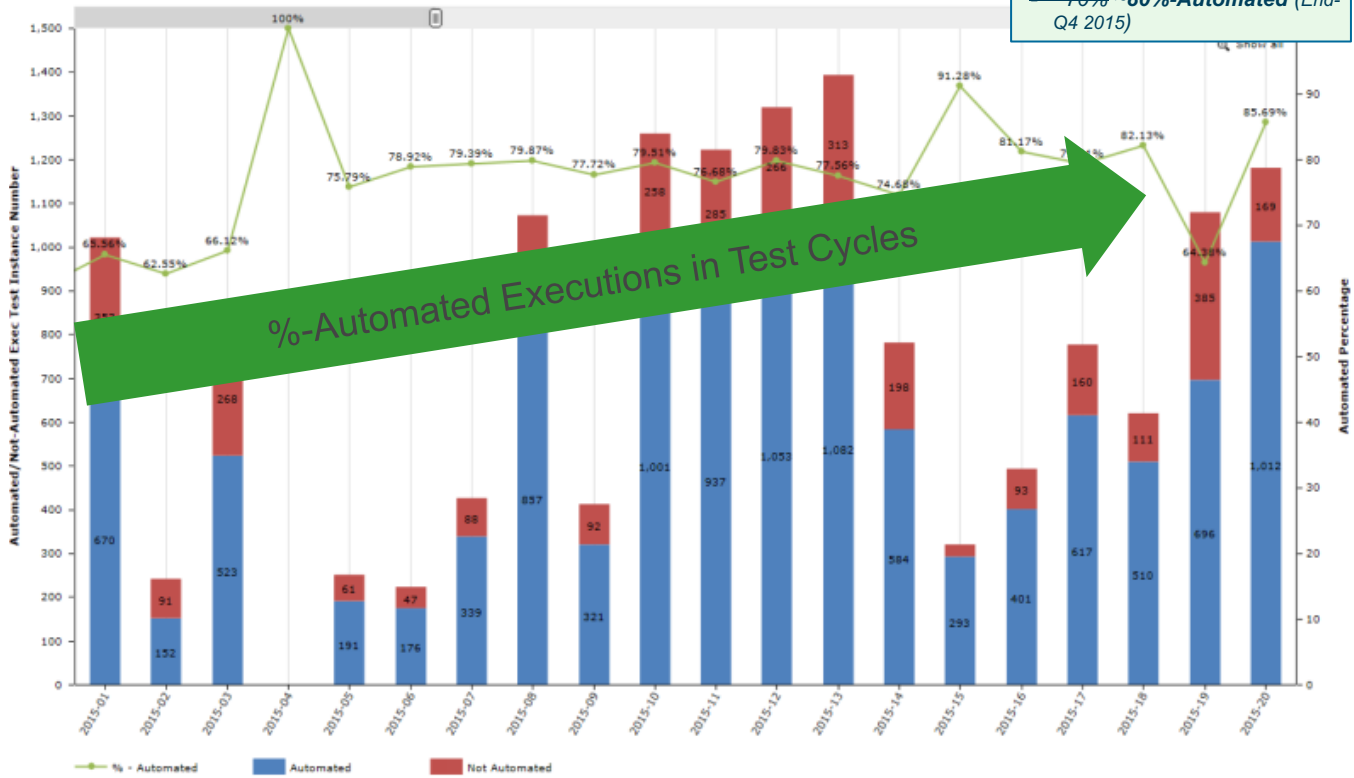
# CTEE AR Summary



# CTE<sup>2</sup> Achievement

Stati  

- 4 DaoCloud
- 51%-Automated (Mar. 2015)
- 53%-Automated (Jun. 2015)
- 66.49%-Automated (Oct. 27, 2015)**
- 70% ~80%-Automated (End-Q4 2015)



# CTE<sup>2</sup> Achievement (Cont.)

Date of snapshot	MRES * QE Owned UTMS Executed Tests	%Automation (Tests in Inventory)	%CTEE-Automation (Actual Executions for Systems & Solutions CTC's)
7/20/2017 (current)	2747	86.17%	42.11% (Cycle-29)* 42.70% (Cycle-27)*
7/14/2017 (current -1)	2737	86.34%	54.79% (Cycle-27)* 52.01% (Cycle-25)*
7/7/2017 (current -2)	2735	86.40%	64.95% (Cycle-27)* 51.92% (Cycle-25)*
6/28/2017 (current -3)	2828	82.07%	51.68% (Cycle-25)* 45.06% (Cycle-23)*

# Summary

- Leverage automation to improve regression efficiency
- Try best to avoid complexity introduced by tool or process
  - light and fast
- A stable continuous execution environment is very helpful for improve execution efficiency
  - ✓ Job scheduling
  - ✓ Automatic execution result upload & issue report
  - ✓ Execution status monitor
  - ✓ Test bed management
  - ✓ Etc ...
- Always try best be lazy (let machine do the job!)





DELL EMC