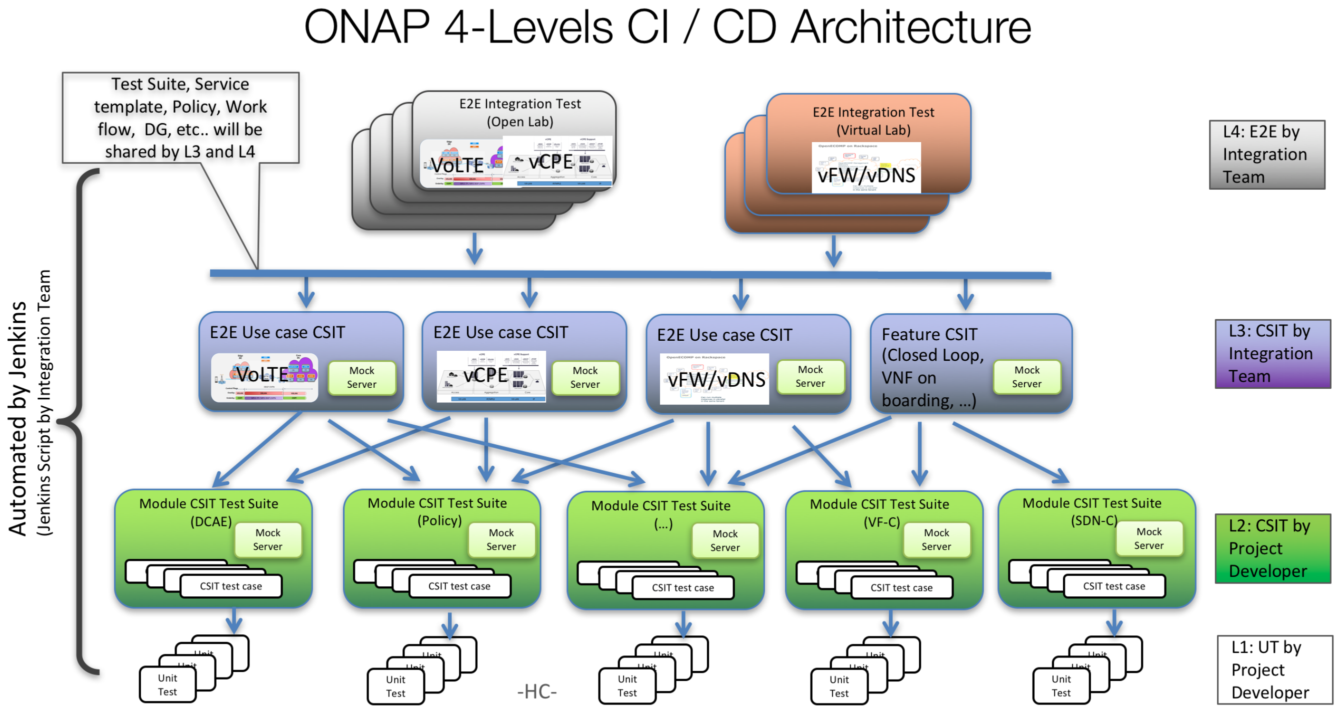
Integration (5/11/2017)

Project Name:

- Proposed name for the project: `Integration`
- Proposed name for the repository: `integration`

Project description:

Integration is responsible for ONAP cross-project system integration, CI/CD, and all related end-to-end release use cases testing with VNFs necessary for the successful delivery and industry adoption of the ONAP project as a whole.



ONAP 4-Levels CI / CD Architecture
Test Suite, Service template, Policy, Work flow, DG, etc.. will be shared by L3 and L4
E2E Integration Test (Open Lab)
VoLTE
vCPE
E2E Integration Test (Virtual Lab)
vFW/vDNS
L4: E2E by Integration Team
Automated by Jenkins
(Jenkins Script by Integration Team
E2E Use case CSIT
VoLTE
Mock Server
E2E Use case CSIT
vCPE
Mock Server
E2E Use case CSIT
vFW/vDNS
Mock Server
Feature CSIT (Closed Loop, VNF on boarding, …)
Mock Server
L3: CSIT by Integration Team
Module CSIT Test Suite (DCAE)
Module CSIT Test Suite (Policy)
Module CSIT Test Suite (…)
Module CSIT Test Suite (VF-C)
Module CSIT Test Suite (SDN-C)
Mock Server
CSIT test case
L2: CSIT by Project Developer
Unit Test
-HC-
L1: UT by Project Developer

Scope:

It provides all the cross-project infrastructure framework and DevOps toolchain (Continuous Integration, etc.), code and scripts, best practice guidance, benchmark and testing reports and white papers related to:

- Cross-project Continuous System Integration Testing (CSIT)
- End-to-End (ETE) release use cases testing with VNFs with repeatability
- Service design for end-to-end release use cases
- (obsolete)Open Lab: building and maintenance of community integration labs
- Continuous Distribution (CD) to ONAP community integration labs
- Reference VNFs that can be used to show how the ONAP platform handles
  - VNF installation
  - VNF life cycle management
  - VNF Requirement compliance

	Category	Primary Contact / Participants	Sub-Category	Description	Problem Being Solved
1	Test	Daniel Rose Christophe Closset Anaël Closson Kang Xi Gary Wu Ran Pollak	Test	• Automated testing infrastructure and tools ◦ CSIT: testing of individual ONAP microservices and small collections of ONAP microservices supported by mocked services as necessary ◦ End-to-End (ETE) test flows using a full ONAP deployment • Code and tools for automatic system testing and continuous integration test flows across ONAP projects • Common guidelines, templates, generic tools, infrastructure, and best practices to help project developers to write unit and system test code • Test requirement from developer point of view.	• Automate the building artifacts/binaries to minimize human errors and reduce engineering costs • Ensure that changes in one project will not break the functionality of other projects • Assure that the entire ONAP project/product functions correctly in the case of continual change in subprojects • Ensure consistency in unit and system testing methodology across all the ONAP projects • Capture security issues
			S3P	• Test cases for performance, scalability, resilience/stress testing, longevity • Benchmarking and performance whitepapers	• Define standard S3P testing metrics • Provide and publish benchmarking results
2	Release	Gary Wu Christophe Closset Anaël Closson	CI Management (ci-management repo)	Scripts and definitions for build and CI jobs in Jenkins ◦ includes any docker build jobs for mock/simulated services ◦ excludes docker build jobs for ONAP components (assumed to be handled by the ONAP Operations Manager project) ◦ Required to support the execution of CI jobs (e.g. for Jenkins)	Required to support the execution of CI jobs (e.g. for Jenkins)
			Autorelease	• Define community-wide artifact versioning and release strategy • Scripts and Jenkins job definitions to build the artifacts/binaries (e.g. zip/tar.gz files) that are used in the release candidates and final release • Detect/resolve cross-project compilation dependency issues • Generate release candidates and final release artifacts	• Detect/resolve cross-project compilation dependency issues • Generate release candidates and final release artifacts
			Distribution (Refer to ONAP Operations Manager project)	• Current decision is to go with docker images as the primary distribution method • Docker builds and images are assumed to be handled by the ONAP Operations Manager project.	(Refer to ONAP Operations Manager project)
			Packaging (Refer to ONAP Operations Manager project)	• Current decision is to forgo any deb or RPM packages, and go with docker images as the primary distribution method • Docker builds and images are assumed to be handled by the ONAP Operations Manager project.	(Refer to ONAP Operations Manager project)
3	Bootstrap	Victor Morales Kiran Kamineni Nathaniel Potter	Bootstrap	A framework to automatically install and test a set of base infrastructure components for new developer	• Reduce the barrier of entry to allow new ONAP developers to ramp up onto active development quickly • Reduce the cost to the community in responding to simple environment setup questions faced by new developers
			Infrastructure Specification	Develop the specifications for the “ONAP compliant” deployment and test environment	Assist the planning and procurement of the necessary hardware and infrastructure for setting up ONAP environments
4	Developer Lab	Yang Xu He Rui Bin Yang	End-to-end release use cases testing with VNFs with repeatability	• Create automatic test cases and script for VF testing • Perform VF compliant testing and verification using tools provided by ONAP • Delivery the testing reports and whitepaper	• Assist define the testing metrics • Reduce adoption risks for end-users
5	E2E Integration Lab	Chengli Wang Kang Xi Yang Xu	End to end deployment in "real" env using Open Lab	• Scripts and definitions for setting up a POC sample deployment of use cases in lab settings • Provisioning, installation, and setup of all the telco equipment such as switches, routers, and gateways to enable end to end testing • Allow remote access to the lab environment for interoperability testing • Automatic updates of code in lab environment from future releases	• Support the needs of consistent, reproducible lab setup for demo and POC purposes • Promote easy interoperability testing with different hardware devices, SDN controllers, etc. • Automate the process of keeping the lab code up to date with the latest changes

6	Reference VNFs Project	Marco Platania Andrew Fenner	Reference VNFs Project	Two basic VNFs, namely a virtual firewall and a virtual load balancer (with virtual DNSs), have been provided. The objectives of the project are to improve, extend and maintain the vFirewall and vLoadBalancer VNFs: • Allow ONAP to change vFirewall rules during execution • Platform independence (Rackspace, vanilla Openstack, Azure, ...) • Visualization tools that allow users to monitor the behavior of the reference VNFs as well as the effect of ONAP closed-loop operations against the VNFs • Tools that allow users to interact with the reference VNFs (e.g. alter the behavior of a VNF so as to violate predefined policies, in order to trigger ONAP closed-loop operations)	• The goal is to build reference VNFs that can be used to show how the ONAP platform manages VNFs installation and lifetime management. • Reference VNFs can also be used as a means to test the platform itself, e.g. verify whether VNFs on-boarding, deployment, and ONAP closed-loop operations work. • Reference VNFS should also demonstrate and document VNF Requirement compliance
7	O-Parent	Gary Wu	O-Parent	• ONAP Parent provides common default settings for all the projects participating in simultaneous release.	• Isolate all the common external dependencies, default version, dependency management, plugin management, etc. • Avoid duplicate/conflicting settings for each project • Each project sets its parent to inherit the defaults from ONAP Parent • Project level external dependencies and versions can be overridden if necessary

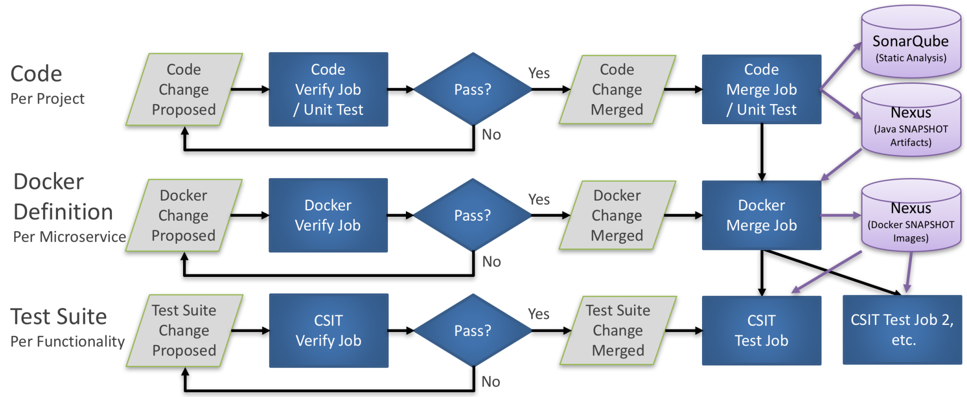
Testing Principles

- We expect test **automation** for all testing in scope for release 1.0

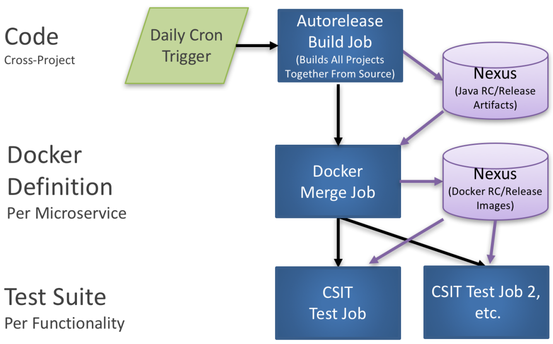
- Regression, Unit and Feature/Function testing should be **triggered by build process**
- All testing must be able to execute on the selected ONAP environments
- Unit Testing for any project should have **at least 30% code coverage**
- Any new feature should be delivered with its associated unit tests/feature tests

Jenkins Testing Flow

ONAP Jenkins Job Flow (/per patch)



ONAP Jenkins Autorelease / RC Job Flow (/daily)

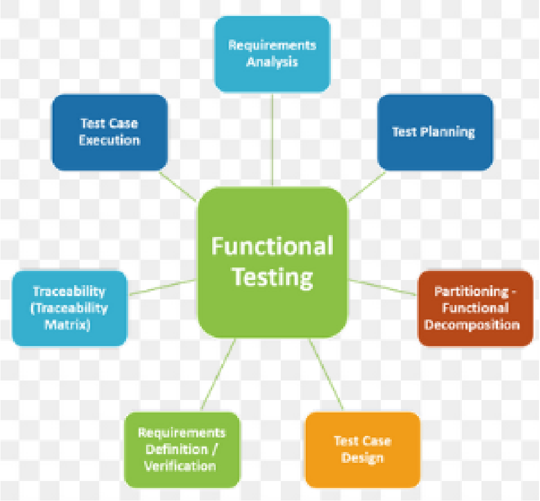


Testing Roles and Responsibilities

Types of Testing	Dev. Team	CSIT Team	E2E Team	S3P Team
Unit Testing	x			
Feature/Functional Testing	x			
Integration/Pair-Wise Testing		x		
End-to-End Testing			x	
Regression Testing	x	x	x	x
Performance Testing				x
Acceptance Testing		x	x	
Usability Testing	x			
Install/Uninstall Testing	x			
Recovering Testing			x	x
Security Testing				x
Stability Testing				x
Scalability Testing				x
Application Testing	x			

FUNCTIONAL & REGRESSION TESTING

Functional Testing - when new features & use cases are introduced that the existing use cases are enhanced to be compatible with the new functionality. For example, if enhancements were needed in the VCPE or VOLTE U/C to upgrade the PNF support introduced in R4 Dublin. ("Dictionary" Definition) *Functional testing* is a software testing process used within software development in which software is tested to ensure that it conforms with all requirements. Functional testing is a way of checking software to ensure that it has all the required functionality that's specified within its functional requirements.


Requirements Analysis
Test Case Execution
Test Planning
Functional Testing
Traceability (Traceability Matrix)
Partitioning - Functional Decomposition
Requirements Definition / Verification
Test Case Design

Regression Testing - when new features & use cases are introduced doesn't break existing use cases. For example, if PNF onboarding/onboarding (PNF package) U/C introduced in R4 Dublin didn't break the functionality of VCPE or VOLTE. ("Dictionary" Definition) *Regression testing* is re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change. Changes that may require regression testing include bug fixes, software enhancements, configuration changes, and even substitution of electronic components. As regression test suites tend to grow with each found defect, test automation is frequently involved. Sometimes a change impact analysis is performed to determine an appropriate subset of tests (non-regression analysis).

Integration Testing - testing which brings together the platform components and new development to realize a use case. For example, testing centered around making sure that the SDC, A&AI, SO, VID, etc components work together to deliver the BBS Use Case. ONAP has a Integration Project, there is a dedicate team who is coordinating the integration efforts. Additional, each of the Use Cases are discussing have "roadmap" their integration work. ("Dictionary" Definition) *Integration testing* is a software testing methodology used to test individual software components or units of code to verify interaction between various software components and detect interface defects. Components are tested as a single group or organized in an iterative manner. After the integration testing has been performed on the components, they are readily available for system testing.

TESTING CONCERNS:

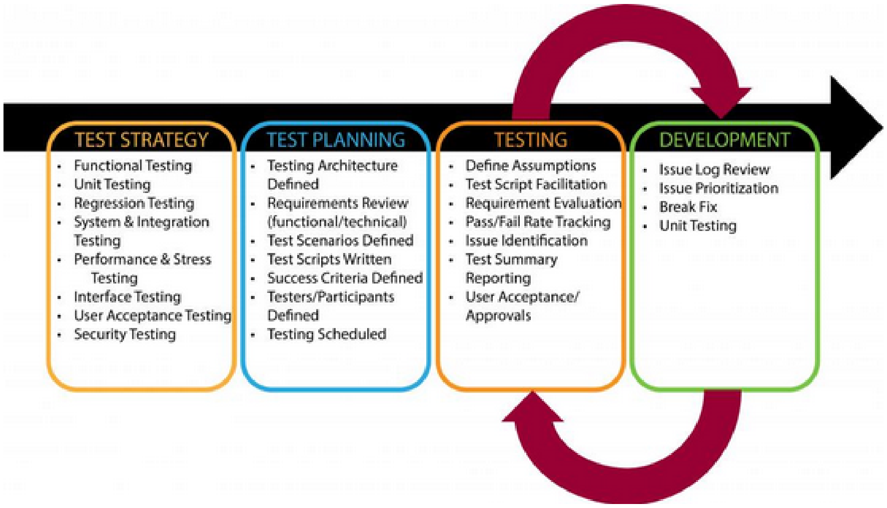
LIMITED RESOURCES - the Integration project (test team) in ONAP is a small team, which will depend on automation as much as possible to try to cover the Regression and Functional testing aspects as much as possible.

RELEASE RISK - there is obviously a risk adding new software without regression and functional testing, in that a lot of integration testing (from R3) was done to make sure that new U/C introduced in R3 worked properly, and now introducing new U/C, functionality, and requirements could potentially break that already tested & working software.

AUTOMATION - it is desirable to have as much automation as possible in testing so that tests can be re-run so that functionality can be re-checked in the current release when new functionality is introduced.

LAB ENVIRONMENT - U/C were developed in particular environments, the xNFs are installed in a particular environment, and these environments change over time. Limitations need to be identified with additional equipment or new equipment etc.

Typical Software Industry Testing Process. ONAP Development & Integration testing can consider and incorporate some of the key concepts with a "typical" software industry testing process to insure quality software is delivered for the Use Case development:



Testing Terminology

- **Unit Testing (UT)** – Unit testing focuses on individual software components or modules. Typically, UT is done by the programmers and not by testers, as it requires detailed knowledge of the internal program design and code; UT may require developing test driver modules or test harnesses. Code coverage is also one of the objectives of UT.
- **Feature/Functional Testing** – Feature/Functional testing, unlike unit testing, focuses on the output as defined per requirement (user story). This type of testing is black-box type testing, geared towards functional requirements on an application basis.
- **Integration/Pair-Wise Testing** – Integration/Pair-wise testing integrates all of the modules of the solution in a single environment to verify combined functionality after integration. It ensures everything comes together and there is end-to-end communications between all the integrated elements.
- **End-to-End Testing** – End-to-End testing involves testing of a complete application environment in scenarios that closely mimic real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems, if appropriate.
- **Regression Testing** – Testing the application as a whole for the modification in any module or functionality. Typically, automation tools are used for regression testing since it is difficult to cover all aspects of the system in a manual fashion.
- **Performance Testing** – Performance testing tests the solution to see what throughput levels can be achieved based on the given platform. Performance testing term is often used interchangeably with ‘stress’ testing (pushes the system beyond its specifications to check how and when it fails) and ‘load’ testing (which checks the system behavior under steady load.). An objective of this testing is to verify that the system meets performance requirements. Different performance and load tools are used for this type of testing. [Note: Capacity testing and performance testing are closely aligned. Capacity testing tests against product requirements to ensure the system meets requirements. Performance testing pushes the system to highest numbers it can achieve before it becomes unstable or success rate is unacceptable.]
- **Acceptance Testing** – Normally this type of testing is done to verify if system meets the customer specified requirements. User or customer do this testing to determine whether to accept application.
- **Usability Testing** – Usability testing is focused on testing of user interfaces (UI’s). It checks user-friendliness along with application flows; e.g., can new users understand the application easily, is proper help documented whenever users get stuck at any point, etc…. Basically system navigation is checked in this testing.
- **Install/Uninstall Testing** – This category of testing validates full or partial install/uninstall, and upgrade and rollback processes on different operating systems under different hardware, software environment including backup/restore mechanisms.
- **Recovery Testing** – This category of testing validates how well a system recovers from crashes, hardware failures, or other catastrophic problems in various configurations such as HA and Geo-Redundancy.
- **Security Testing** – Security testing looks for vulnerabilities in the software that would make the system susceptible to malicious users. It tests how well the system is protected against unauthorized internal or external access. It checks if system and databases are safe from external attacks.
- **Stability Testing** – This type of testing validates that the system can run in steady state for an extended period of time without any system downtime or crashes. Typical stability test is run for 72hrs. If any problems are encountered during the stability test, such as application crash, high failure rates, the test is stopped and issue is investigated. The stability test is restarted after a fix is identified.
- **Scalability Testing** – This type of testing is an extension of performance testing. The purpose of scalability testing is to validate that the system can scale efficiently by identifying major workloads and mitigate bottlenecks that can impede the scalability of the application.
- **Application Testing** – Tests the platform in the context of a particular application.

Architecture Alignment:

- How does this project fit into the rest of the ONAP Architecture?
  - What other ONAP projects does this project depend on?
    - All ONAP projects
- How does this align with external standards/specifications?
- Are there dependencies with other open source projects?
  - Robot
  - Jenkins
  - OpenStack
  - Docker

Other Information:

- **Preliminary V1 Plan**
    - **Integration V1 Release Plan**
- **ONAP Community Labs Specification**
    - **Lab Resource**
- **Link to seed code (if applicable)**
    - ECOMP existing repos:
        - testsuite
        - testsuite/heatbridge
        - testsuite/properties
        - testsuite/python-testing-utils
        - demo
        - ci-management
    - OPEN-O existing repos:
        - integration
        - ci-management
        - oparent
- **Vendor Neutral**
    - This project is vendor neutral
- **Meets Board policy (including IPR)**
    - Yes

Use the above information to create a key project facts section on your project page

Examples of end to end use cases:

- VoLTE

    - VoLTEservice design and creation: VNFs, network resource, workflow, alarms, DGs, DCAE template are onboarded. VoLTE e2e network service, including the connectivity between data centers, are designed and distributed successfully

    - Close loop design: DCAE threshold crossing rules, Holmes correlation rules, and operation policy are designed and distributed by CLAMP successfully

    - Instantiation: VNFs are deployed in edge and core data centers successfully, underlay and overlay WAN network cross the two data centers is setup correctly

    - Auto-healing/auto-scaling: With event data from VIM or VNF, certain auto-healing/auto-scaling policy is triggered and proper actions are taken to remedy the situation

    - Service termination: After user terminates the VoLTE service, VNFs and related resource for the service are properly removed, and WAN connectivity between the two data center is also removed successfully.

- Residential vCPE use case

    - Design time: All the VPP-based VNFs are successfully onboarded.

    - Design time: Infrastructure and per-customer service are created. Associated workflows, policies, DGs, and data analytics created, validated, and properly distributed

    - Instantiation: Infrastructure service is instantiated and properly configured. This is performed only once.

    - Customer order: Per-customer service is instantiated and properly configured. This is performed on-demand.

    - Data plane: Once a customer service is up and running, packets can be exchanged between the customer BRG emulator and the webserver.

    - Auto-healing: Inject a packet loss event to invoke threshold crossing event, which then causes APPC to restart the vG_MUX VM. Service is back to normal once restart is complete.

Key Project Facts

Project Name: Integration

JIRA project name: integration

JIRA project prefix: integration

Repo name:

- integration
- demo
- testsuite
- testsuite/heatbridge
- testsuite/properties
- testsuite/python-testing-utils
- ci-management
- oparent

Lifecycle State: incubation
Primary Contact: Helen Chen helen.chen@huawei.com
Project Lead:
mailing list tag [integration]
Committers:

See above

*Link to TSC approval:
Link to approval of additional submitters: