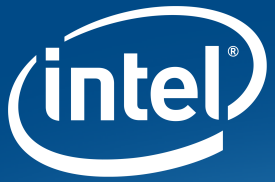


Intel's Journey to Build Quality In:

How QA and Test Automation Drive DevOps

MANISH AGGARWAL

Legal Disclaimer

General Disclaimer:

Technology Disclaimer:

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Performance Disclaimer:

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Agenda

- Introduction
- What we do
- Software delivery challenges
- Where we started
- Solution Discussion: Design principles and best practices
- Where we ended up
- Q&A

About Myself

- Software Engineering Manager at INTEL, Austin, TX
- 19+ years of software design/development experience in Telecom and Networking
- Working at INTEL for 9 years in Data Center Group (DCG)

"I believe that quality products lead to happy customers, and happy customers keep companies in business."

What we do!

- Deliver System on Chip (SoC) to the top tier wireless telecom vendors in the world
- So whenever you use your Smartphone, watch a YouTube Video, or communicate over an LTE network you are probably using some hardware or software that we built!

- Note: Don’t waste time comparing iPhone X and Galaxy 8. Buy both ☺

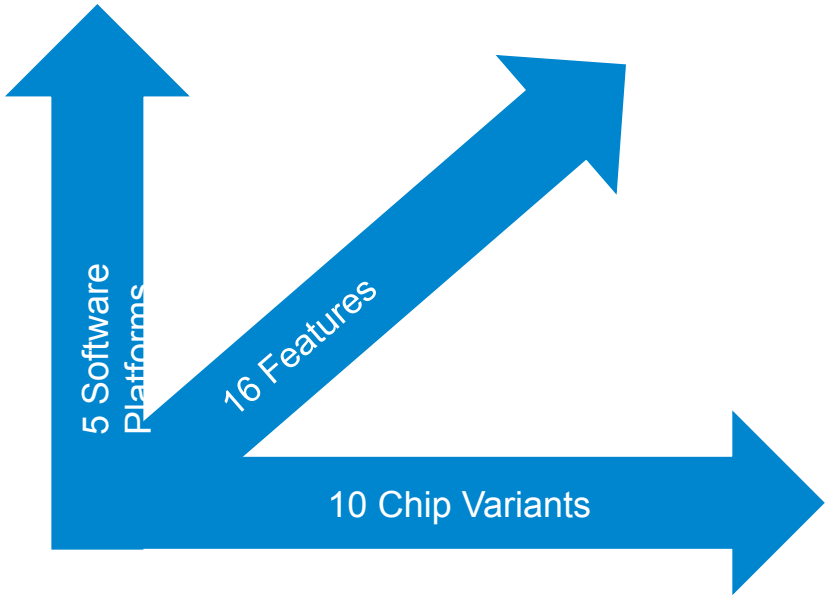
Why is Quality Assurance So Difficult?

QA and Test Automation – why so difficult for DevOps?

- The middle child of your pipeline, gets the least love/visibility
- Last check point in the release cycle. Absorbs delays.
- It is actually hard
- Ever increasing Test matrix
- Embedded device config that cannot be easily updated
- Traditional manual testing is risky :
  - Slow and Error prone
  - Repetitive ; Monotonous
  - Subject to Delays, re-work and unpredictability
  - Test engineers commonly scramble to navigate between the pace of Dev, and the requirements of Ops around environments, compliance, security, and more.

What was our challenge?
20,000+ Test Matrix, and growing!

Business need to verify the Hardware part for a wide range of combinations.



Hundreds of Tests for each Engine

The Test Matrix Grows Exponentially
Why?

- ***Legacy!!*** *Many years of baggage!*
- ***Backward Compatibility:*** *Deliver new features without breaking any old functionality*
- ***New Flavors always keep pouring!*** *It never ends*
- ***Need to decouple dependency between team/flavors:*** *to be able to release independently*
- ***Time to market driven by large OEM/device manufacturers :*** *the customer is asking for a release sooner than planned. Can we do that without de-scoping any features!*
- ***How much testing is enough?***

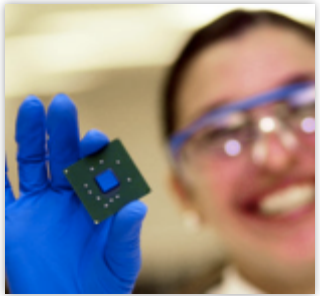
4
Releases/Year

DAYS
QA Lead Time per CI Cycle

85%+
Unknown Code coverage

3M
Lines of Code

100's
cases run/day



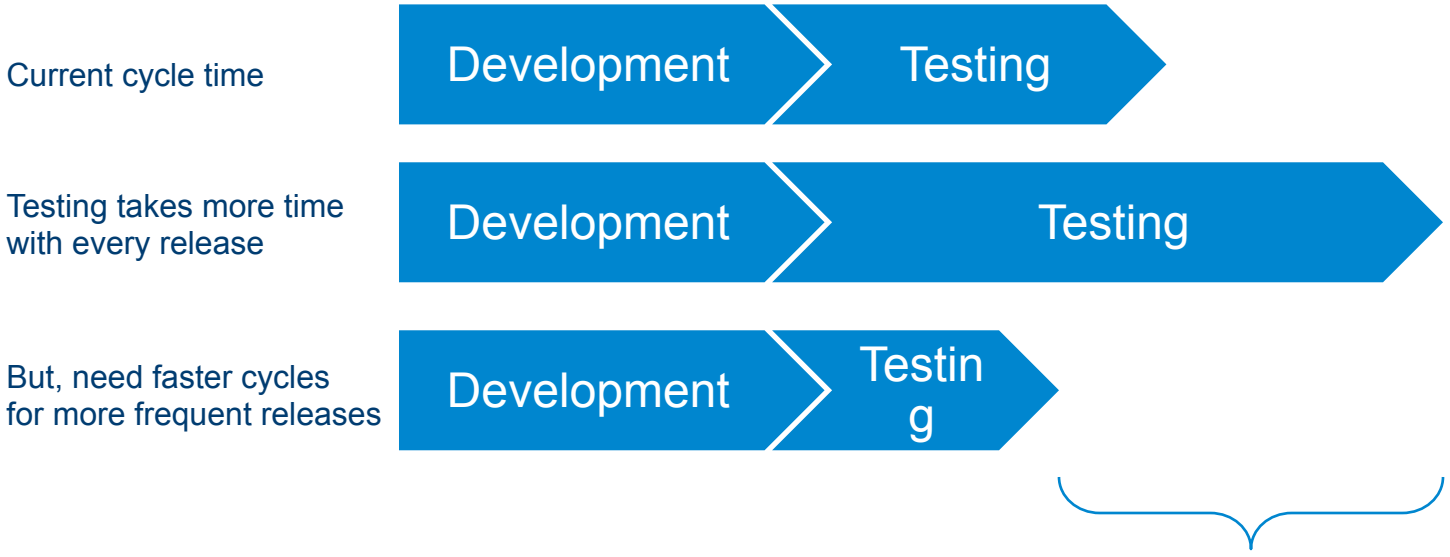
96
Permutations of S/W and H/W per product

Where we started

How it looks on the ground:

- ***It worked on my machine!***
- **Flaky/inconsistent test:**
  - This test was working till yesterday. Not sure why is this failing now.
  - How to troubleshoot? Can you go back to an older build and check when did this test last work?
- **Show me the numbers! What's our status?**
  - Customer is asking for code coverage numbers and memory leak testing. BTW, did I tell you that due to budget constraints, we have a strict hiring freeze this year.
  - How many tests do we have?
  - How many tests are passing? How soon can we make a release?
- **No additional headcount**

Time to Market (TTM) or Quality?



Options:

- Increase cycles times and delay TTM?
- Shorten cycle times and risk quality?
- **Test smarter and faster with automatio**

What we want to achieve:

1. Testing at scale: growing matrix, limited resources
2. Consistency (repeatability)
3. Visibility
4. Known Quality & status
5. Metrics drive decisions
    - Elevate and share metrics dashboards and our status across the organization
    - Keep everyone in Sync

How do you support 20,000 Test Cases!!

Our Solution: Design Principles

1. **Create test scripts that are re-usable**
   - Parameters to the rescue!
2. **On-demand is key:**
   - Tests are stored in on-demand **self-service repository** for engineers to access
   - On-demand **environment provisioning/ configuration**
3. **Easy Access:**
   - Users **access the repository** the way they prefer: using UI, CLI, Script/API, triggered automatically from a pipeline task, etc.
   - Can choose existing test to re-use or customize
   - Can run set of test cases manually or on a schedule.
4. **Provide shared visibility that's easy to consume:**
   - High level status dashboards
   - Drill down capabilities/alerts

How ElectricFlow from Electric Cloud Helped?

- ElectricFlow forms a single, flexible, scalable platform for us to design around
- All tests automated using ElectricFlow
- Custom UI for accessing tests across the org
- Results automatically emailed to stake holders
- Based on test results the pipeline automatically proceeds: approval gates, change requests, and consequent pipeline stages are triggered.
- Shared reports and dashboards provide visibility

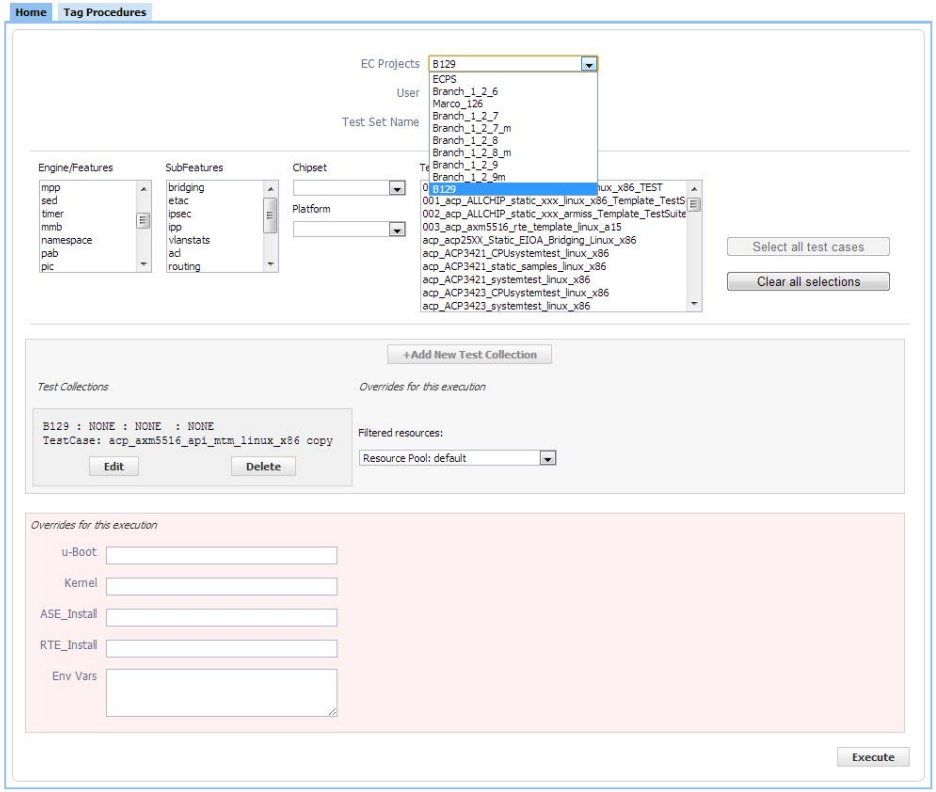
1. Re-usable, Parameterized Test Automation Scripts

- Object-oriented approach:
  - Each tier in the test scripts is **parameterized**: Jobs, Procedures, Releases, Teams etc.
- Each test is categorized/labeled with these parameters, for **easy grouping and sorting**:
  - Release Branch
  - Engine/Feature
  - Sub Feature
  - Chip Type
  - Platform
  - Test Type
- Procedures supply information for data-driven execution
  - Create lists from the data in properties (e.g. Engine, Subfeature, Chipset, Platform)
  - Default operational parameters, overridden during automation
  - This way, we create dynamic and variable-length option lists

2. On-demand Self-service Tests Repository

- This vastly improves the maintainability!
- Users can view all tests and procedures based on search criteria/filters that they care about.
  - The procedures visible are lot more manageable.
  - Users can save their customized test-lists.
  - Users can refer to the test-lists saved by other users.
- Users easily choose test to re-use or modify – using UI, CLI, Script/API, triggered automatically from a pipeline task, etc.
- Users can select tests to run:
  - executed as-is.
  - Modified
  - executed with different builds
  - Executed using different resources
- Everyone share the same tests! Developers can run the QA tests, against their sand-box, even before check-in!!

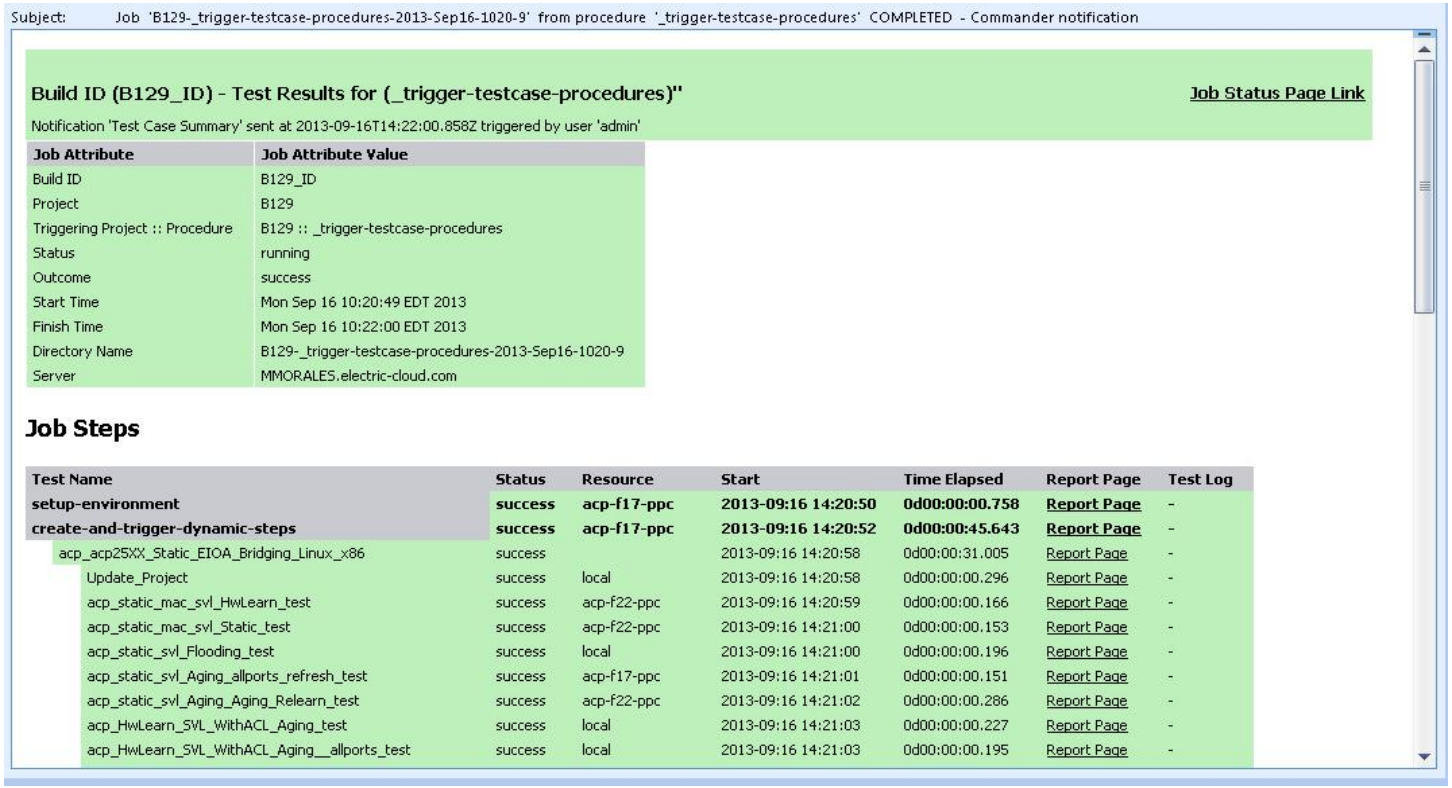
Accessing the catalog:



Accelerated Feedback Loops

- Color-coded email notification sent to tester for reviewing / troubleshooting.
- Notifications for certain critical tests (Regression) are sent to an expanded list of recipients, including Management.
- Notifications have all the information needed for troubleshooting (links to logs, test results, etc.)

Email Notification



Subject: Job 'B129-_trigger-testcase-procedures-2013-Sep16-1020-9' from procedure '_trigger-testcase-procedures' COMPLETED - Commander notification

Build ID (B129_ID) - Test Results for (_trigger-testcase-procedures)" Job Status Page Link

Notification 'Test Case Summary' sent at 2013-09-16T14:22:00.858Z triggered by user 'admin'

Job Attribute	Job Attribute Value
Build ID	B129_ID
Project	B129
Triggering Project :: Procedure	B129 :: _trigger-testcase-procedures
Status	running
Outcome	success
Start Time	Mon Sep 16 10:20:49 EDT 2013
Finish Time	Mon Sep 16 10:22:00 EDT 2013
Directory Name	B129-_trigger-testcase-procedures-2013-Sep16-1020-9
Server	MMORALES.electric-cloud.com

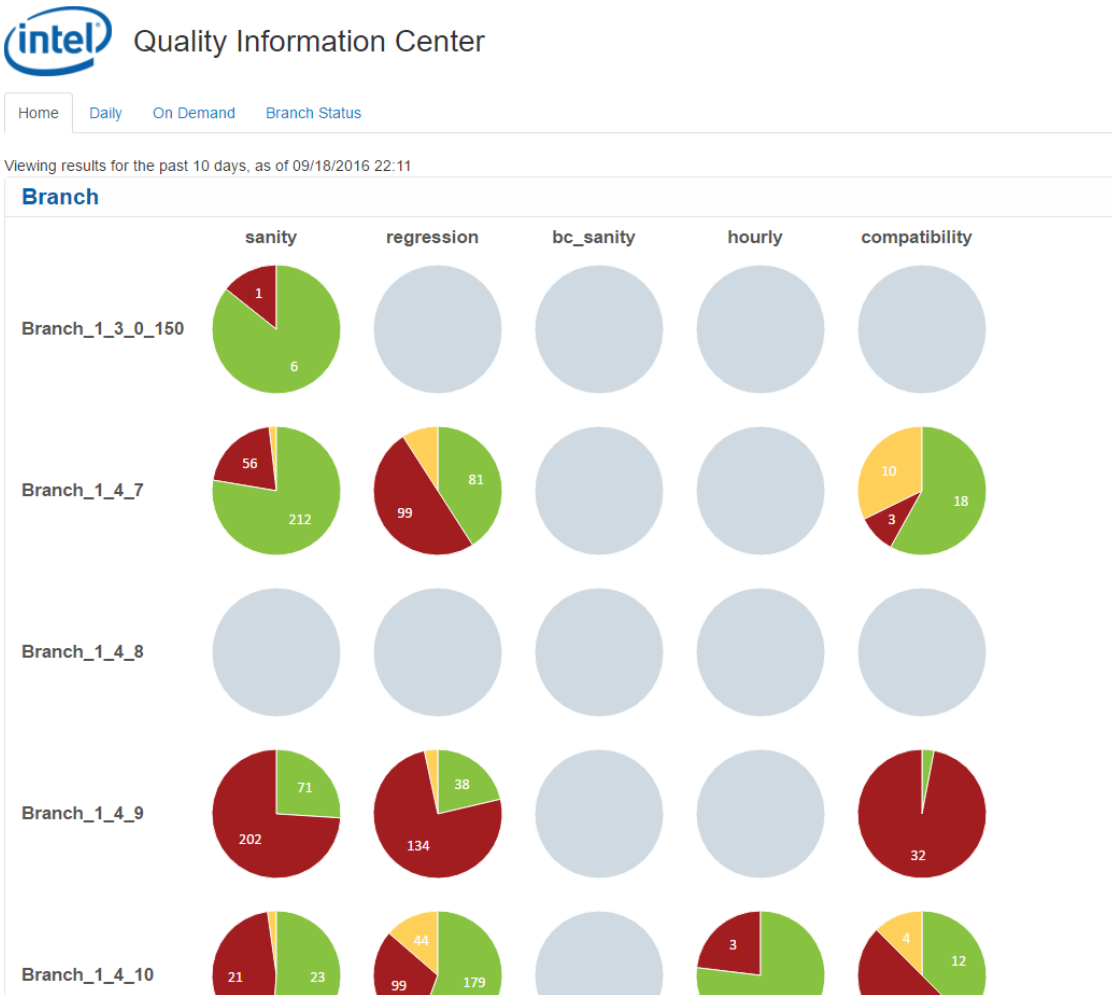
Job Steps

Test Name	Status	Resource	Start	Time Elapsed	Report Page	Test Log
setup-environment	**success**	**acp-f17-ppc**	**2013-09-16 14:20:50**	**0d00:00:00.758**	**Report Page**	-
create-and-trigger-dynamic-steps	**success**	**acp-f17-ppc**	**2013-09-16 14:20:52**	**0d00:00:45.643**	**Report Page**	-
acp_acp25XX_Static_EIOA_Bridging_Linux_x86	success		2013-09-16 14:20:58	0d00:00:31.005	Report Page	-
Update_Project	success	local	2013-09-16 14:20:58	0d00:00:00.296	Report Page	-
acp_static_mac_svl_HwLearn_test	success	acp-f22-ppc	2013-09-16 14:20:59	0d00:00:00.166	Report Page	-
acp_static_mac_svl_Static_test	success	acp-f22-ppc	2013-09-16 14:21:00	0d00:00:00.153	Report Page	-
acp_static_svl_Flooding_test	success	local	2013-09-16 14:21:00	0d00:00:00.196	Report Page	-
acp_static_svl_Aging_allports_refresh_test	success	acp-f17-ppc	2013-09-16 14:21:01	0d00:00:00.151	Report Page	-
acp_static_svl_Aging_Aging_Relearn_test	success	acp-f22-ppc	2013-09-16 14:21:02	0d00:00:00.286	Report Page	-
acp_HwLearn_SVL_WithACL_Aging_test	success	local	2013-09-16 14:21:03	0d00:00:00.227	Report Page	-
acp_HwLearn_SVL_WithACL_Aging__allports_test	success	local	2013-09-16 14:21:03	0d00:00:00.195	Report Page	-

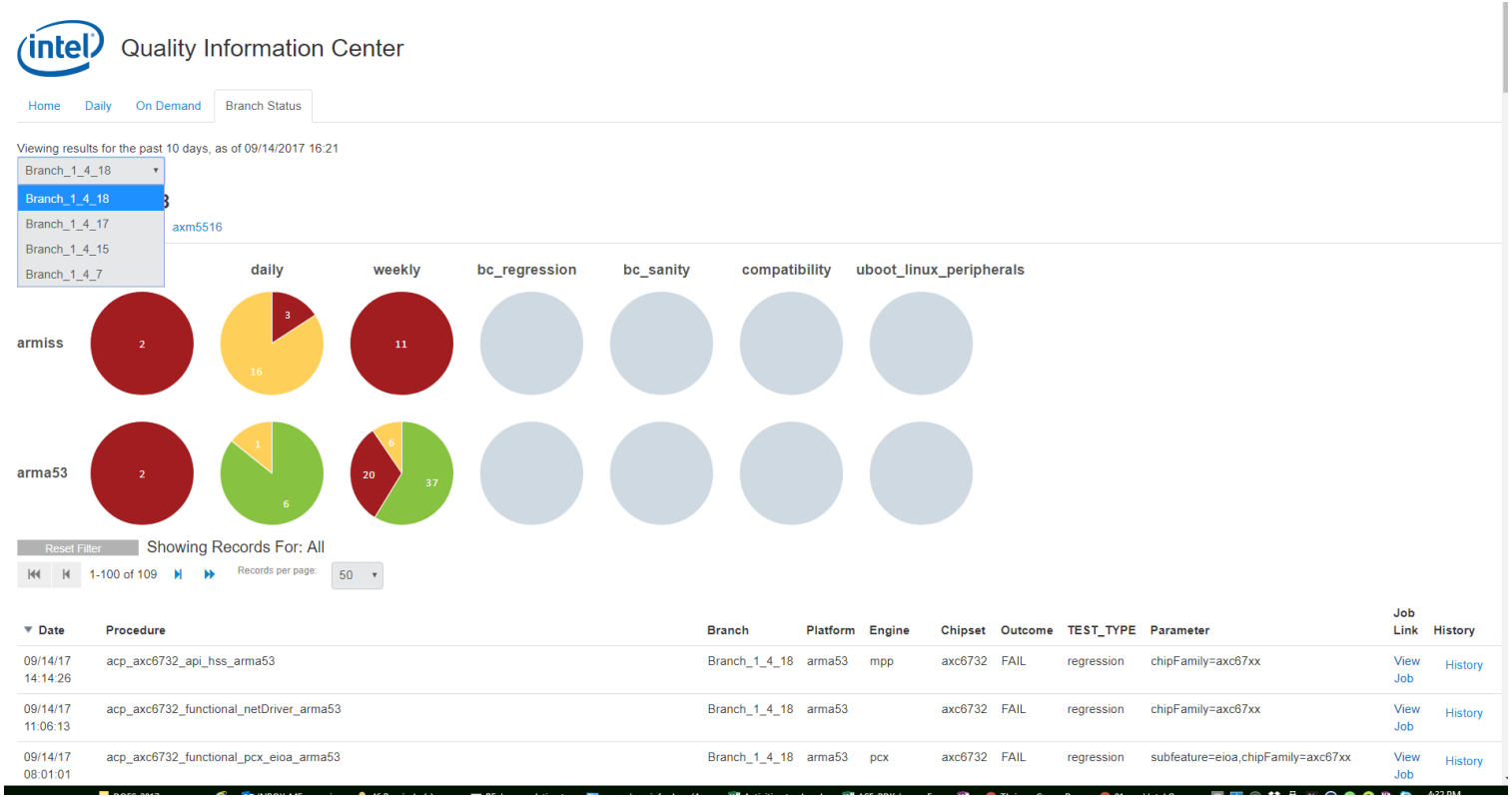
Shared Visibility

- **Bird's eye view is critical + the ability to drill down**
- **Shared language around metrics and status, along with ability to troubleshoot**
- Reports and Dashboards provide at-a-glance visibility to all stake holders re: the health and quality of the release and critical milestones.
- Easy to consume charts for high-level visibility (no need for drill-down data for non-engineers/management)
- Reports are used heavily internally as a gauge for the business, and determine release-readiness
- Data drives the business- we all converge on these metrics

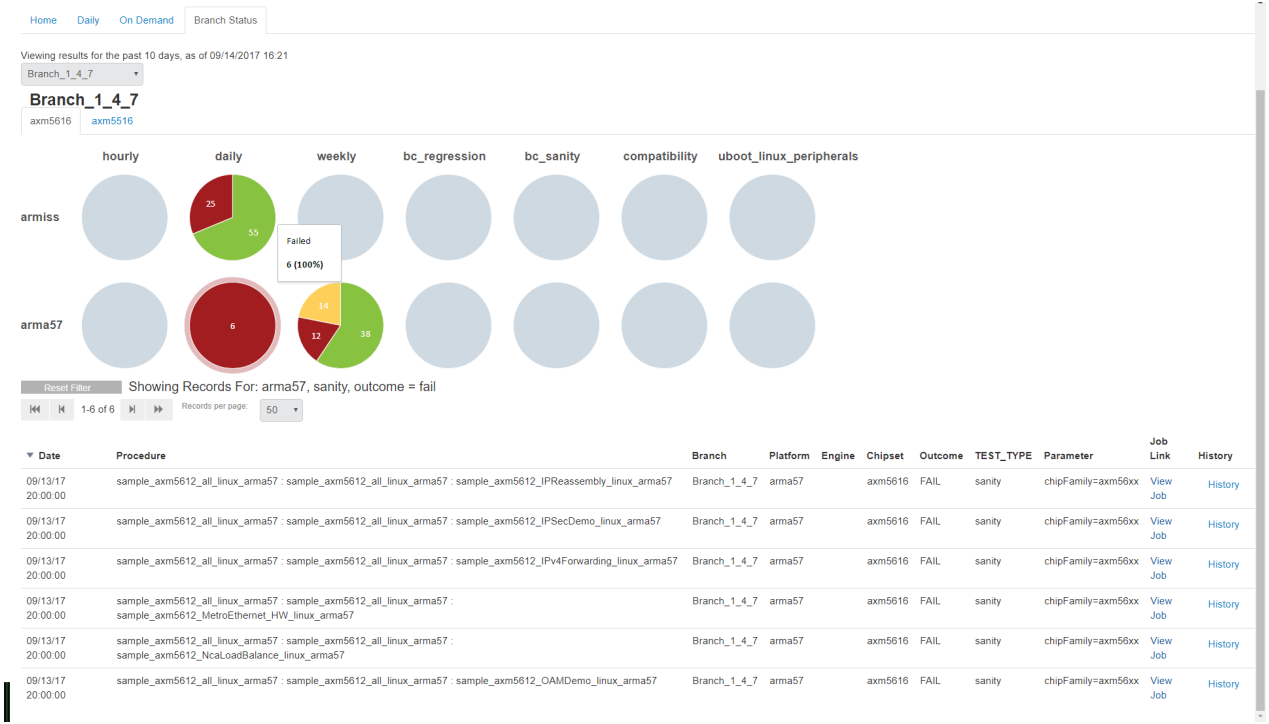
Major Release



Specific release/platform health

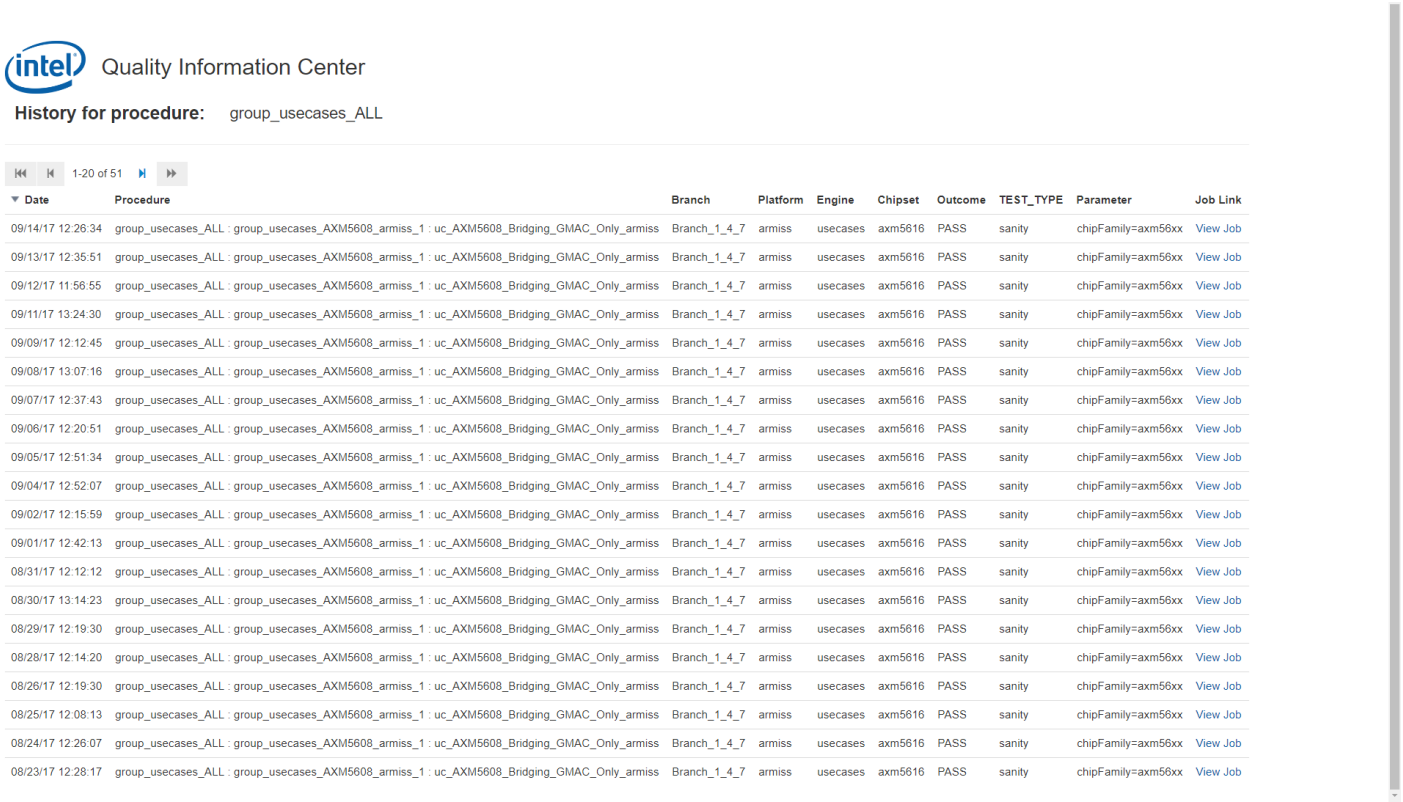


What tests are failing?



Date	Procedure	Branch	Platform	Engine	Chipset	Outcome	TEST_TYPE	Parameter	Job Link	History
09/13/17 20:00:00	sample_axm5612_all_linux_arma57 : sample_axm5612_all_linux_arma57 : sample_axm5612_IPReassembly_linux_arma57	Branch_1_4_7	arma57		axm5616	FAIL	sanity	chipFamily=axm56xx	View Job	History
09/13/17 20:00:00	sample_axm5612_all_linux_arma57 : sample_axm5612_all_linux_arma57 : sample_axm5612_IPSecDemo_linux_arma57	Branch_1_4_7	arma57		axm5616	FAIL	sanity	chipFamily=axm56xx	View Job	History
09/13/17 20:00:00	sample_axm5612_all_linux_arma57 : sample_axm5612_all_linux_arma57 : sample_axm5612_IPv4Forwarding_linux_arma57	Branch_1_4_7	arma57		axm5616	FAIL	sanity	chipFamily=axm56xx	View Job	History
09/13/17 20:00:00	sample_axm5612_all_linux_arma57 : sample_axm5612_all_linux_arma57 : sample_axm5612_MetroEthernet_HW_linux_arma57	Branch_1_4_7	arma57		axm5616	FAIL	sanity	chipFamily=axm56xx	View Job	History
09/13/17 20:00:00	sample_axm5612_all_linux_arma57 : sample_axm5612_all_linux_arma57 : sample_axm5612_NcaLoadBalance_linux_arma57	Branch_1_4_7	arma57		axm5616	FAIL	sanity	chipFamily=axm56xx	View Job	History
09/13/17 20:00:00	sample_axm5612_all_linux_arma57 : sample_axm5612_all_linux_arma57 : sample_axm5612_OAMDemo_linux_arma57	Branch_1_4_7	arma57		axm5616	FAIL	sanity	chipFamily=axm56xx	View Job	History

When did this last pass?



Quality Information Center

History for procedure: group_usecases_ALL

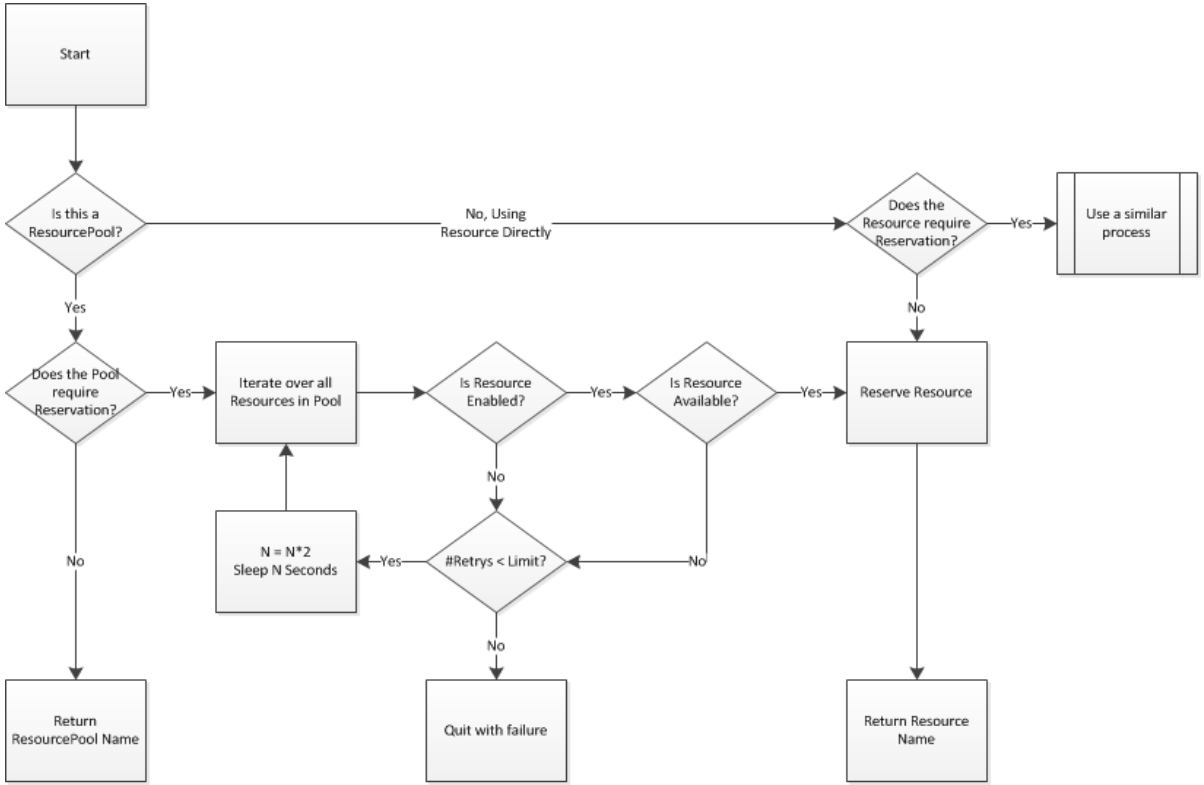
1-20 of 51

Date	Procedure	Branch	Platform	Engine	Chipset	Outcome	TEST_TYPE	Parameter	Job Link
09/14/17 12:26:34	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/13/17 12:35:51	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/12/17 11:56:55	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/11/17 13:24:30	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/09/17 12:12:45	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/08/17 13:07:16	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/07/17 12:37:43	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/06/17 12:20:51	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/05/17 12:51:34	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/04/17 12:52:07	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/02/17 12:15:59	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
09/01/17 12:42:13	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
08/31/17 12:12:12	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
08/30/17 13:14:23	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
08/29/17 12:19:30	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
08/28/17 12:14:20	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
08/26/17 12:19:30	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
08/25/17 12:08:13	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
08/24/17 12:26:07	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job
08/23/17 12:28:17	group_usecases_ALL : group_usecases_AXM5608_armiss_1 : uc_AXM5608_Bridging_GMAC_Only_armiss	Branch_1_4_7	armiss	usecases	axm5616	PASS	sanity	chipFamily=axm56xx	View Job

On-demand is key

- For both executing test
- And provisioning infrastructure for tests to run!!
- Fully automated spin up/down – no more waiting for IT to provision
- No more drift configuration- consistent throughout all environments
- Shared resources improve utilization

Hardware Reservation Process



Bonus: More tests, better quality, less work!

- The framework was easily extended (thanks to parameters) to allows re-use of the same tests for:
  - Code coverage
  - Memory Leak
  - Binary Compatibility Verification

Keep in mind:

- Divide and conquer
- Continuous improvement – can’t boil the ocean, iterative process of optimization
- Keep focus on the end goal – don’t let the intermediate setbacks de-motivate the team.
- Every code can have bugs. Yes, even automation!

Where we ended up:

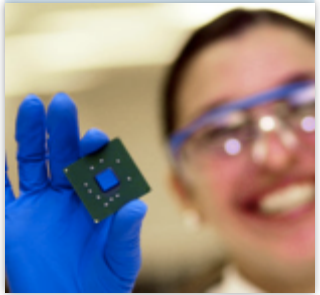
100
Releases/Year

10 min
QA Lead Time per CI Cycle

87%+
Known Code coverage

4M
Lines of Code

15K
Test cases run/day



96
Permutations of S/W and H/W per product

Before … After

Benefits

- Helped manage the enormous amount of work, without increasing head-count.
- Allowing frequent SW deliveries to our customers,
- Continuous monitoring of the health of the product
- More time at hand for test development
- Clear handle on the Quality of the SW that is being shipped out of the doors
- Shared language and metrics: Quality is everyone's responsibility- and now we all know where we stand!

THANK YOU

Q&A