Venkatesh Prasad

Venkatesh.prasad@ikea.com

Abstract

Projects working on Releases may need to update the daily status of the Defects, Test Cases and Stories to their respective stakeholders involved in the release.   
When the work starts for a release, there are frequent changes made to the release artifacts and updating the status of these changes to the management is challenging. Manual ways of generating reports in excel sheets may consume lot of time and efforts of project resources. There could also be human errors and data manipulations from project teams and management may have wrong status of the release. To overcome these problems, we can use Live CLM Dashboards to project live status of test cases, defects, stories and requirements. Reports based on Current and Historical Data can be visualized using Pie, Bar, Linear graphs and table format reports.

Reporting with Collaborative Lifecycle Management

Release wise Test and Defect status reporting using CLM Dashboards

**Pre-Requisites and Assumptions :**

This document is relevant and applicable to only those projects who have their project artifacts created in Collaborative Life Cycle Management tool, i.e

Rational Team Concert used for managing Stories, Tasks, Defects, Epics (Features)  
Rational Quality Manager used for managing Test Cases, Test Scripts, Test Plans, Test Case Execution Records, Test Case Results and Test Suits  
Rational Doors NextGen used for managing Collections, Modules, Files, Business Diagrams, etc

We also assume there is Jazz Reporting Services installed on the CLM Server which has the Report Builder and Data Collection Components installed

**Purpose :**

Projects involving several teams will have to work in sync to achieve a common goal like Release or Sprint. A typical project working on Releases involves Development Teams, Test Teams, SMEs, Architects, Design Team, Requirements Analysis Team and Managers working together to complete a Release which may last from 2 weeks to 2 months.

When the work starts for a release, there are frequent changes made to the release artifacts and updating the status of these changes to the management is challenging. Manual ways of generating reports in excel sheets may consume lot of time and efforts of project resources. There could also be human errors and data manipulations from project teams and management may have wrong status of the release. To overcome these problems, we can use Live CLM Dashboards to project live status of test cases, defects, stories and requirements. Reports based on Current and Historical Data can be visualized using Pie, Bar, Linear graphs and table format reports, which can be populated into a single CLM Dashboard.

These dashboards have the capability to display cross project reports aswell where it can pick up data from multiple project areas from all applications (RTC, RQM, RDNG) and display reports.

There could also be projects who may not follow the scrum processes and may have Kanban concept implemented in their project, in which case, the CLM dashboards will still hold good for reporting purposes.

**Dashboards :**

Reports in CLM can be built in two ways and populated in clm dashboards

1. Using inbuilt pre defined reports along with queries
2. Using Report Builder

We have considered the example of IKEA project where the project works on releases. Project has several test teams and reporting the test status to the stakeholders was a challenge. This was done earlier through 20+ pages of PPT document which was circulated twice a day through emails to 100+ stakeholders.

To avoid this manual process, to save time, to get live status of the test executions, and to get overall test and defect status, we worked through a solution and implemented CLM dashboards.

A dedicated Project Area is created in RTC for reporting purposes, where each Release is considered a separate Team Area in RTC. Reports created for those releases are populated in their respective Team Area’s dashboard.

**Automation of daily LCR reporting via CLM.**

Daily LCR (Launch Control Room) reports which were sent via ppt are stopped and CLM dashboards are created and used for LCR reporting

**Test Reports currently available:**

* Accumulated Test Status : Provides overall test status for each team.
* Test Burndown : Provides count of pending tests to be executed
* Test Increment : Provides test status based on test increments (test suits)
* Feature wise test status : Provides featurewise test status
* Test Status for Master/Child Test Plans : Provides test status of all the child test plans belonging to the master test plan, in a single graph

**Defect Reports currently available:**

* All Open Defects by Severity
* All Open Defects by Status
* Detailed view of the above 2 queries
* All open Tasks

**Challenges**

1. CLM report builder is a light weight tool with some limitations.
2. Setting up the right data in CLM for the reports to fetch right results. Setting up the right data will include creating the defects, stories, tasks, test artifacts and requirements and establishing the traceability between them by linking the artifacts, and also frequently updating the artifacts with the latest statuses.
3. Following the process to have right links between the artifacts.
4. Proper planning of test execution to achieve proper burn down graph. If the scope changes happen, it will reflect in the burndown graphs as ups and downs instead of burning down
5. Defects should be linked to features and test cases

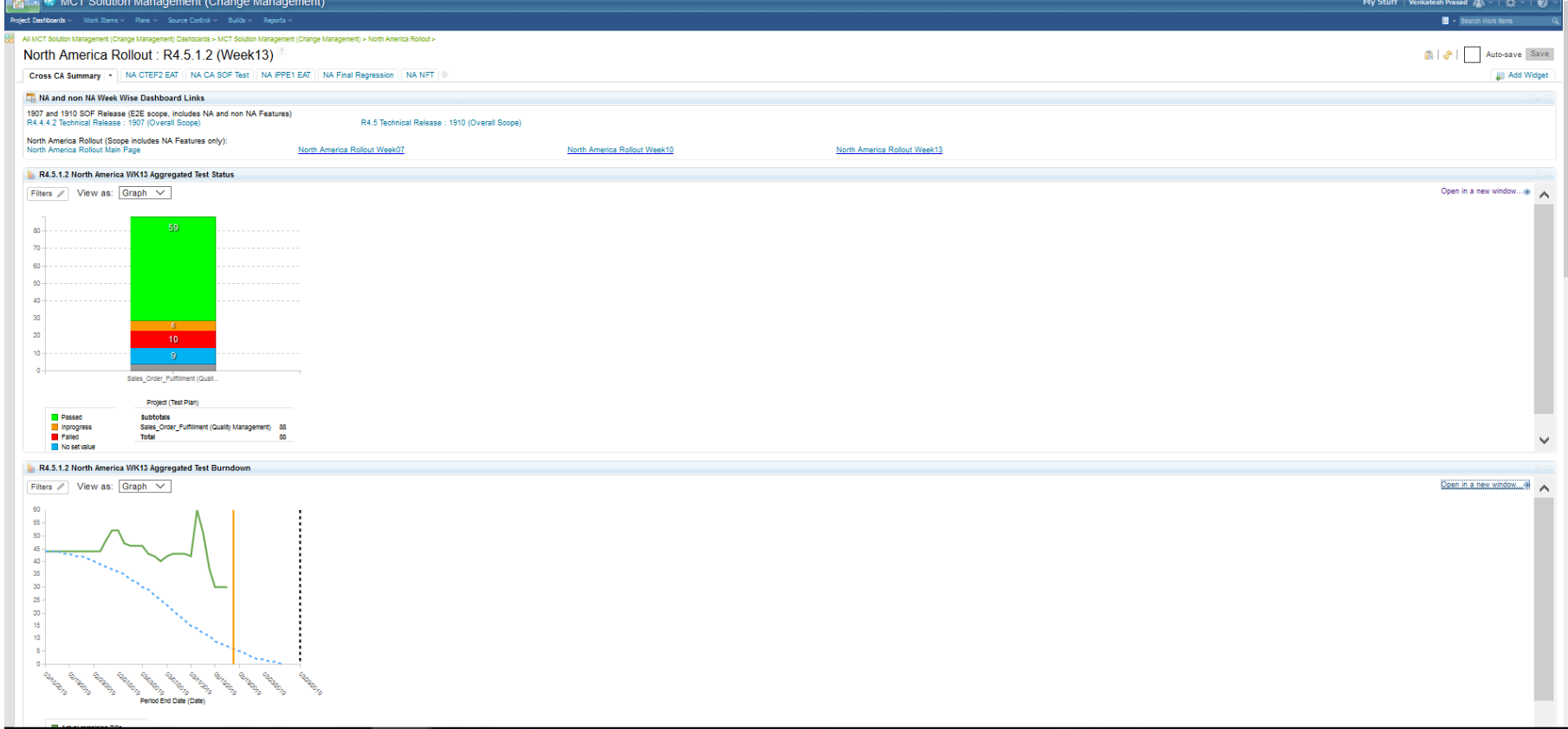
**Report Requirements and Limitations**

1. We would require execution record (TCER) to be created in the beginning itself and kept in the No Run status for all the test cases to provide correct graph.
2. Limited function available for sql query and results. For example date format cannot be changed in the output (mmddyyyy).
3. Planned curve cannot be configured daywise in test burndown graph. By default there is only S curve, Curve in and curve out options
4. Sync between CLM database and Jazz Reporting Services database for historic trending graphs cannot be set more than twice a day as per recommendations from IBM Rational product support. Any changes made to test artefacts after this sync job, will not appear in the graph. This may show mismatch data between Accumulated graph and burndown graph
5. Loading of graphs will be slow depending on the number of Items displayed in the graph.

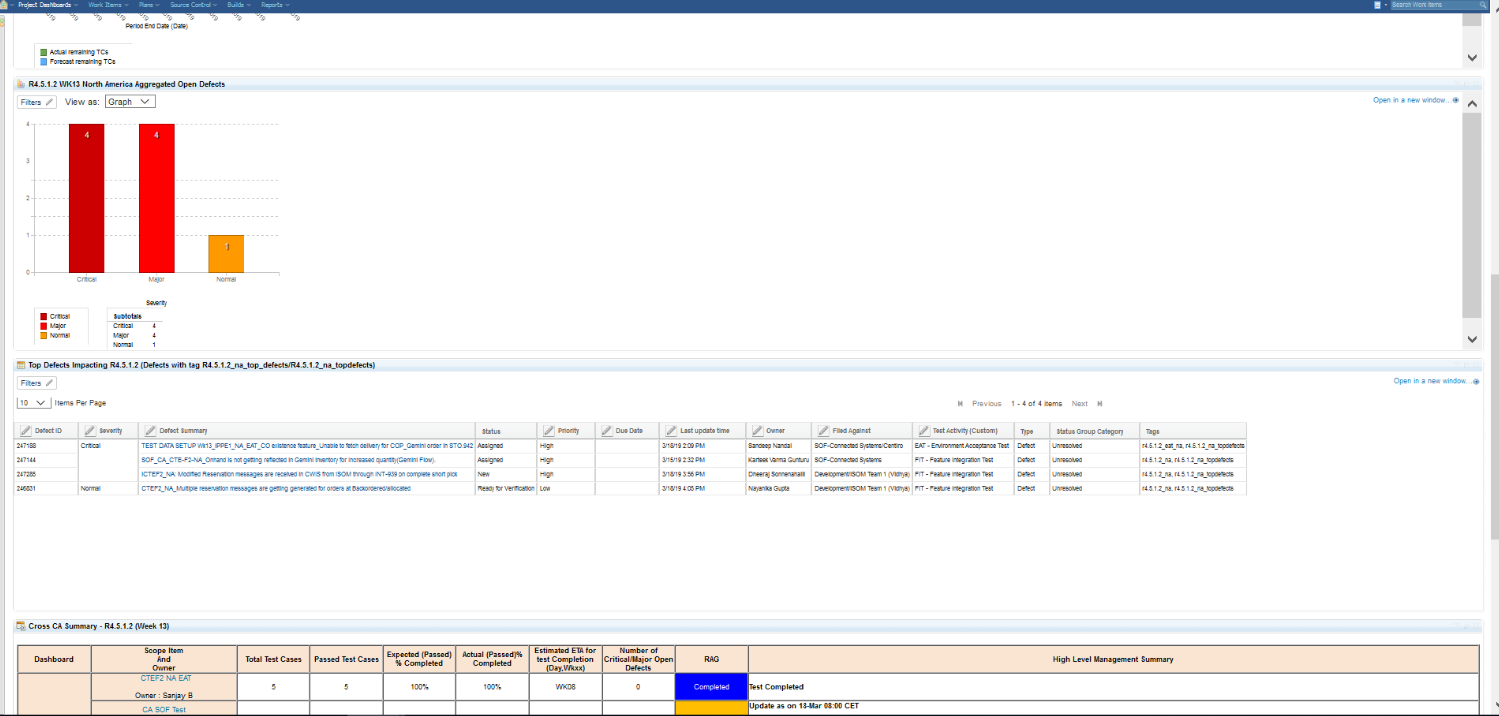
**Business Impact**

* To capture Regression test cases and results, either they should be linked to the existing feature or a new regression feature should be created in that release. Suggestion has been accepted and new Regression Feature is created for tagging regression test cases
* All not passed test cases feature wise – May be required before go live.
* Defect Trending Graph. This is one of the KPIs and a screen shot of it is given below
* List of Test Cases without TCER
* List of defects without feature linked
* Graph Names :
* 1) Severity wise FIT Open Defects - by Feature
* 2) Severity wise Functional (BT and SIT) Open Defects - by Feature
* 3) Severity wise All Open Defects - by Feature
* 4) Severity wise All Open and Closed Defects - by Feature
* 5) Defects with RCA
* 6) Defects without RCA
* 7) Team wise Open Defects - by Feature
* 8) Defect Aging
* 9) Defect burn-down

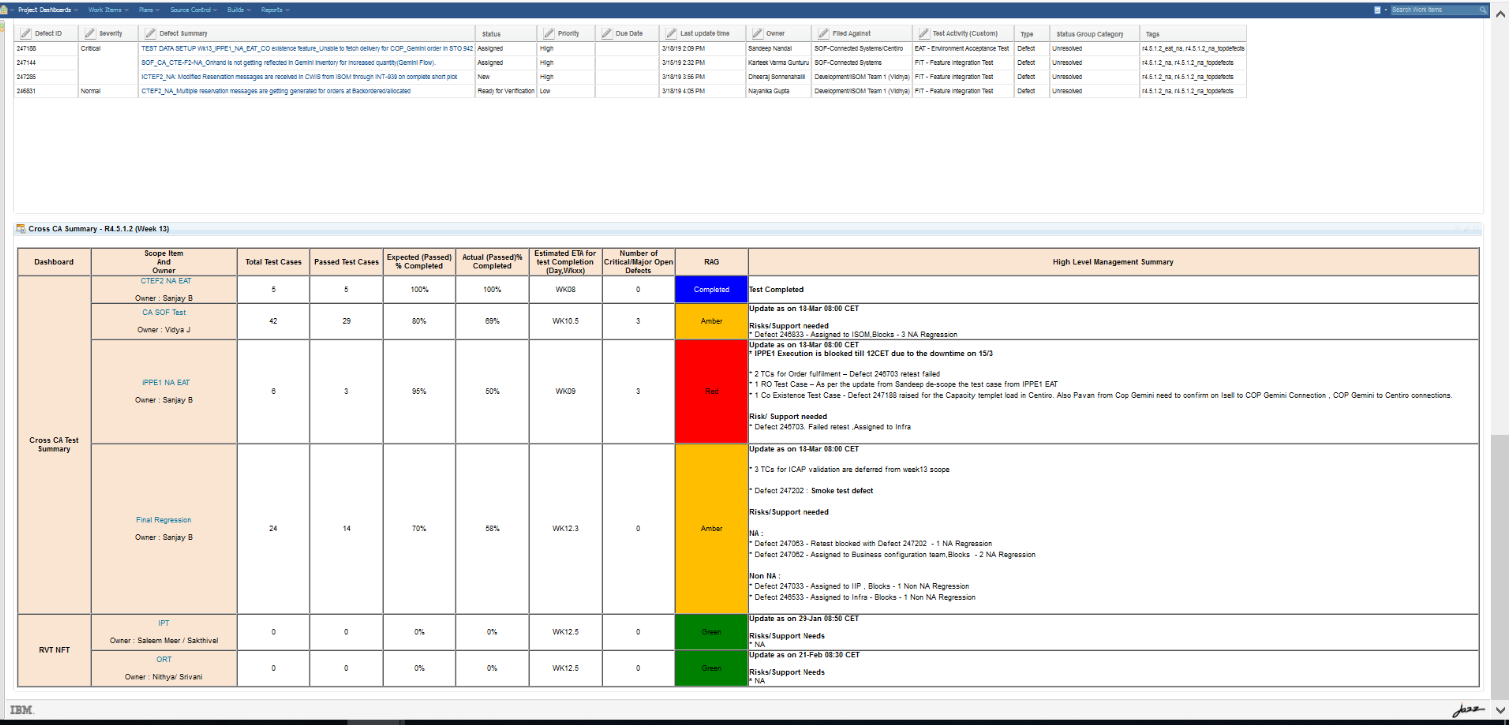
Overall release test status bar graph (current live data) and overall release test burndown graph (historic data which refreshes twice a day)



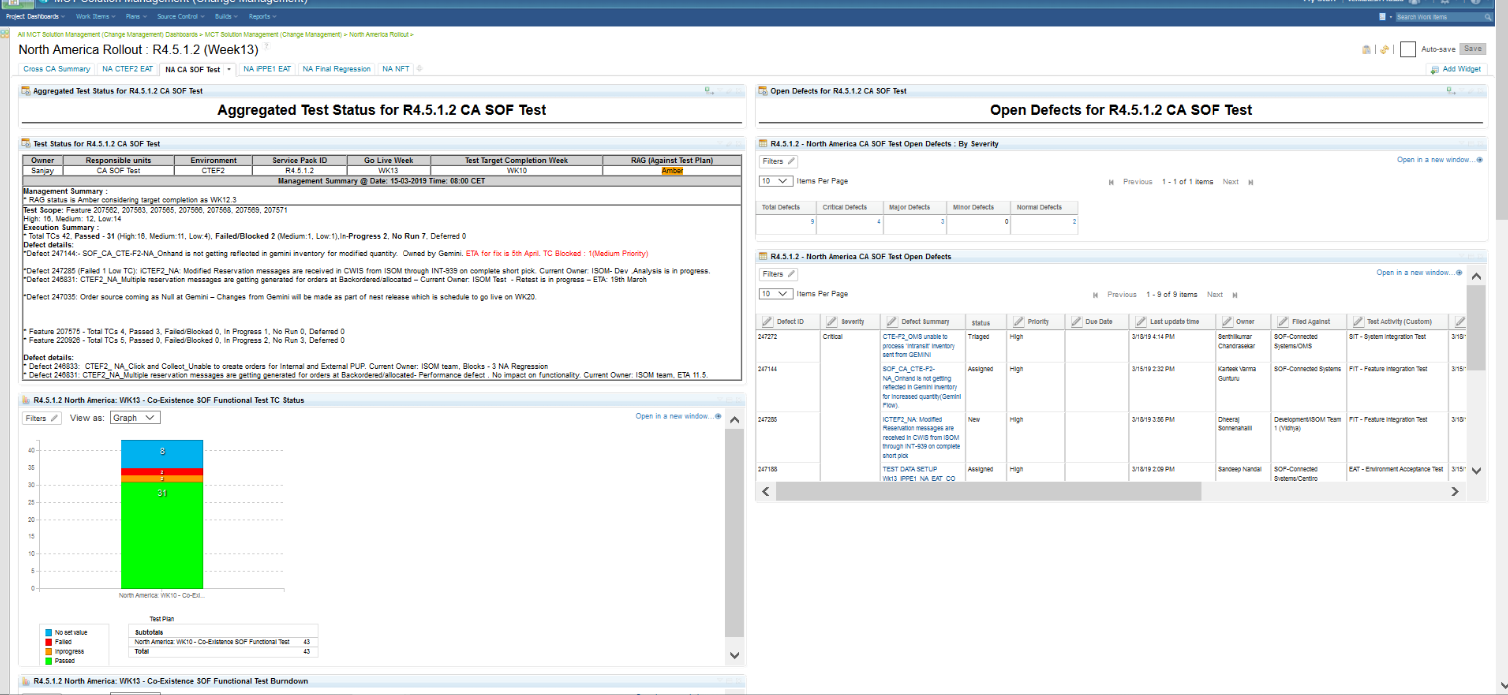
Overall release defect status (bar graph) and top defects impacting the release (table type report)



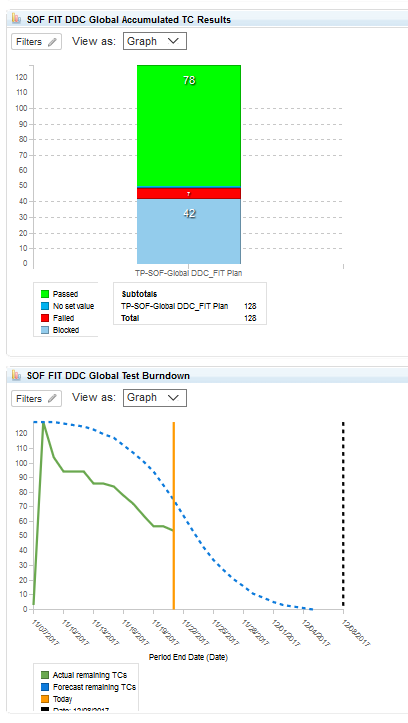
Overall highlevel summary of all teams participating in the release



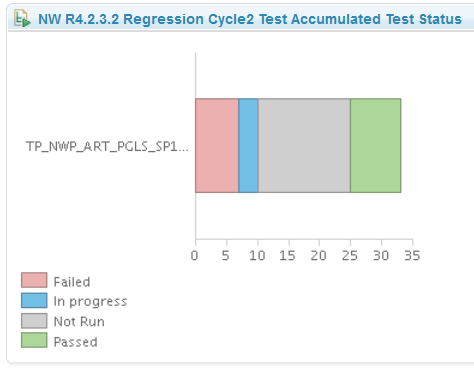
Individual Team’s Test and Defect status along with team’s summary.



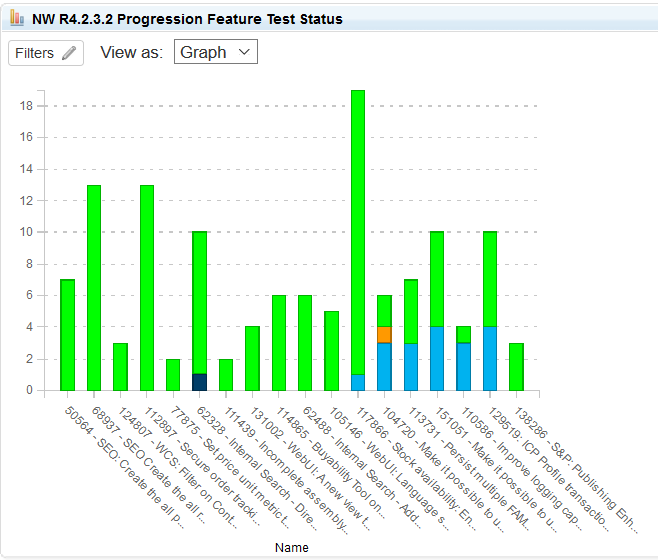
These are Report Builder graphs



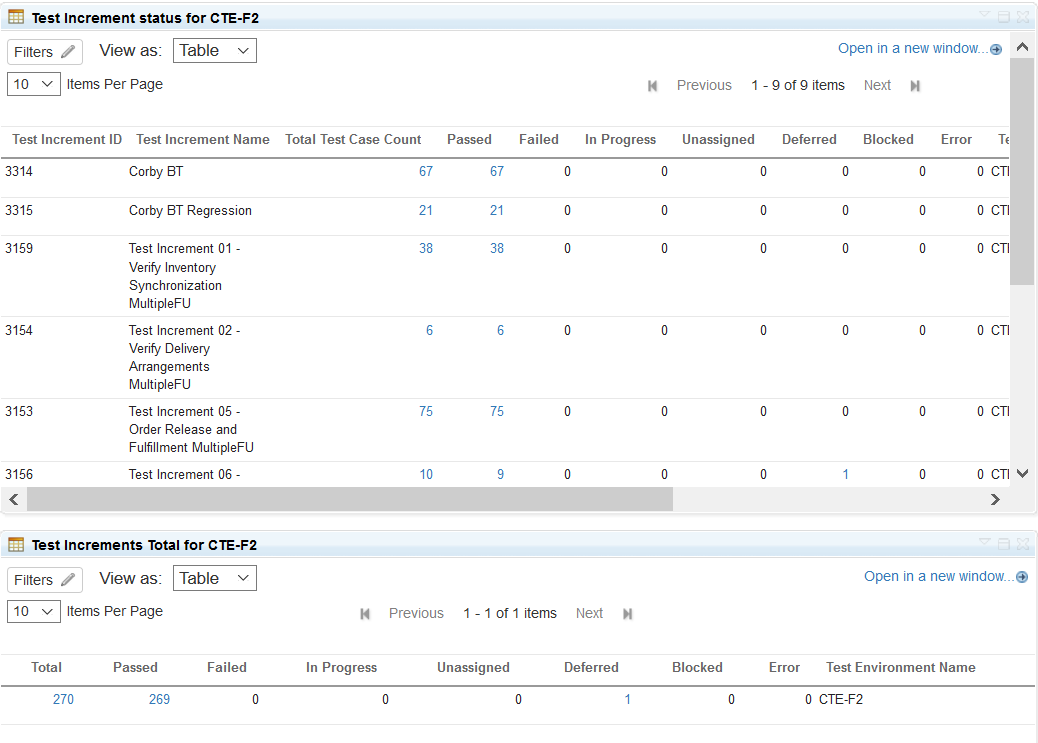
This is a graph generated with the inbuild reporting feature available in RQM

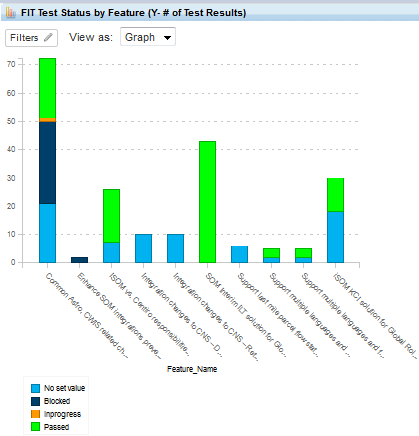


Report builder graph showing test case status in each feature (x-axis is features and y-axis is number of test cases)

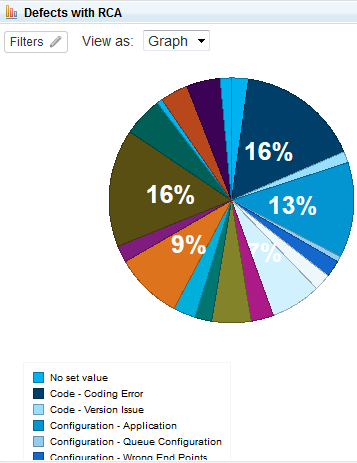


Report built in Report builder using Test suites in RQM

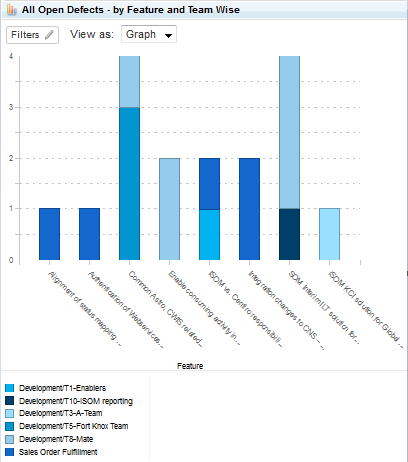




Pie chart showing defect based on root cause analysis type



Defects Bar graph based on features segregated team wise



Bar graph showing all open defects with severity segregated feature wise

