

The following factors need to be considered before creating the Test Execution Run Plan.

Interdependency of test cases – Critical path

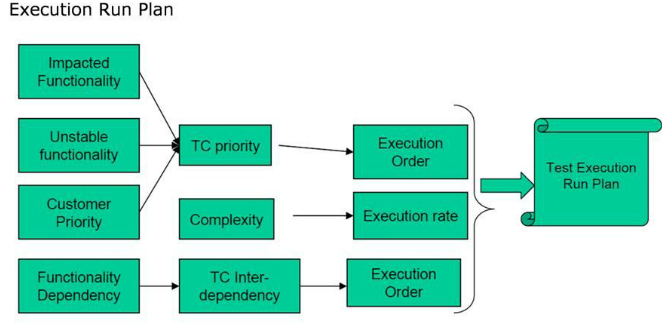
In terms of functionality

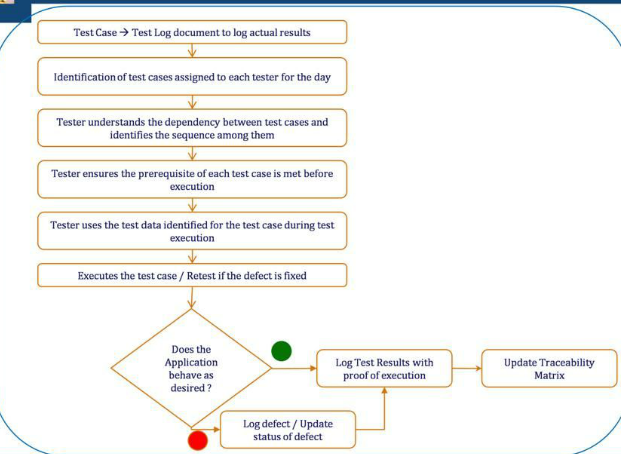
In terms of period specificity

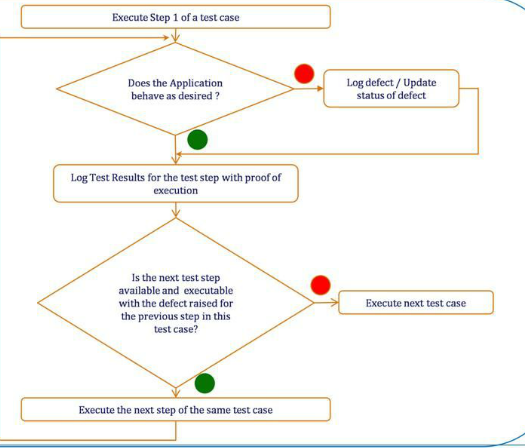
Knowledge level of testers

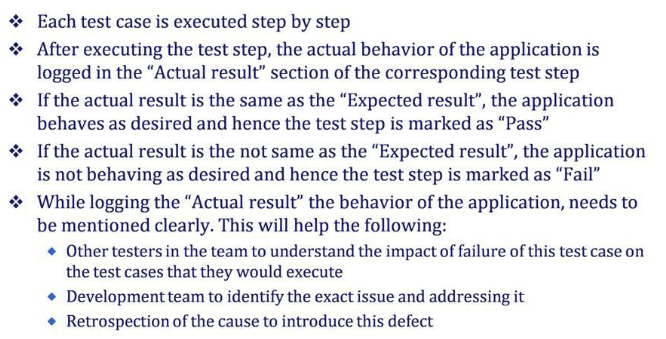
Complexity / Size of test cases

Planned Multiple Builds

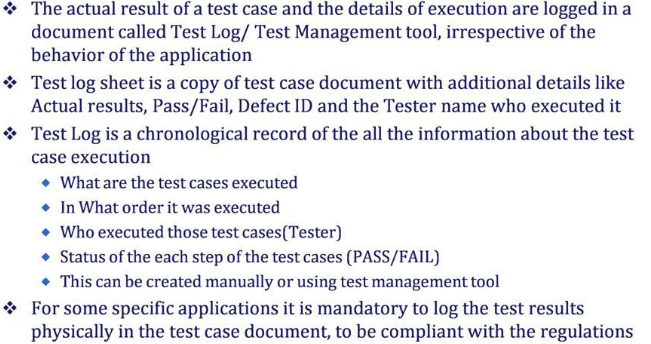












Regression Testing is yet another important phase in the test execution cycle. This testing is done to ensure that the application is intact and works as expected and the defect fix process has not introduced any new defects in the system. Increase in the number of issues in the regression tests indicate an unstable system. A set of regression test cases are identified in the beginning of the test execution cycle- which are run typically after decided time where in most of high severity defects are fixed. Regressions test suites can either run manual or can be executed using automated test suites depending on the needs and the timelines associated with each project.

Retesting of defects is yet another important aspect and phase after the defects are fixed. We must ensure that every defect is retested before the status of the test case is updated/changed. This will help us bring in either new issues or even may be reopen the same issue- of the defect has not been fixed. In a nut shell all the related modules will need to be tested. End result of the retesting phase will lead to the change in state of the defect.

**Test Reporting Guide-** There are various factors that need to be considered before we prepare a final test report

Structure of Testing: We have to schedule your test sequentially or parallel according to the project needs

Reproduce: We must ensure the failed test cases/defects are tested multiple times

Isolate: We must ensure we test various combinations and then come to a conclusion on the nature of the defect

Generalize: We need to test it another browser-zero in on the exact cause of the defect

Compare: We must compare and analyze similar test cases

Summarize: We must summarize and portray the findings in detail with supporting data

Condense: We must trim unnecessary and unintended information

Disambiguate: We must remove ambiguity in reporting by using clear words so as to not create a confusion in the minds of the developers and other testers working on the defect

Neutralize: We must be fair and impartial- make sure we document all the facts in detail

Review: We must ensure all the above points are taken care of in spirit and word so as to deliver a neat and detailed test report

**Why do we need to track the test execution activity.** This is a very important aspect of the test execution process as a whole. We need to keep all the stakeholders well informed about the current status and what is being planned for future so the remaining part of the cycle. The tracker should consists of information throwing light on:

**Areas of concern**

Time taken to execute the test cases

Defect spread across the various modules of the application by their severity and criticality

Defect prone areas of the application

Impact of these defects on the time to release

Progress and status of testing

**There are various factors that are measured frequently(on a daily as well as weekly basis):**

A reality check to know the project's health and status

To present a picture to the stakeholders

**Detail of the factors:**

Number of test cases planned Vs test cases executed: As per the project plan, we would have planned to execute certain X number of test cases for a given day/week. However there are times when we would be able to exceed the number(owing to the simplicity of the test cases/lack of issues) or could result in a lesser number of test cases being executed(owing to environmental downtime/open issues/clarifications)

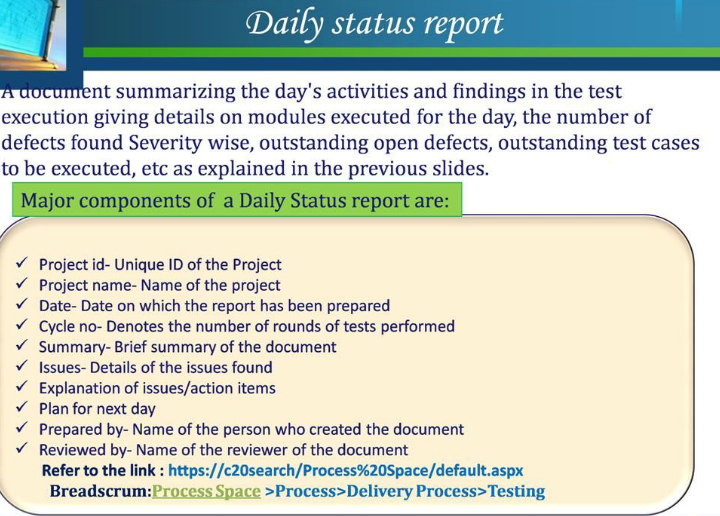
Number of test cases Passed: Details of the number of test cases in “PASS” state

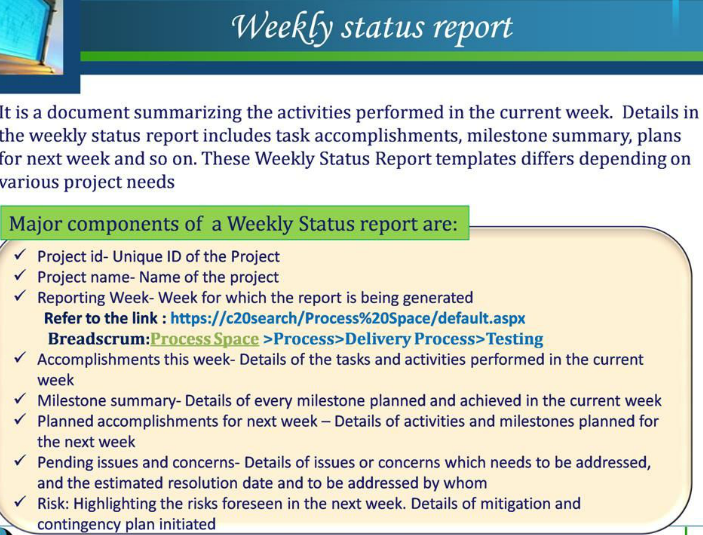
Number of test cases Failed: Details of the number of test cases in “FAIL” state

Outstanding Defects: This line item would track the details of all the outstanding defects based on their

Severity

Module





Here are some of the metrics that will be calculated with the help of the information we collected in terms of test cases and defects making the status reporting more meaningful. Explanation of each in detail:

1. Defect Ageing- Defect Age (in Time) is the difference in time between the date a defect is detected and the current date (if the defect is still open) or the date the defect was fixed (if the defect is already fixed).

Formula: Defect Age in Time = Defect Fix Date (OR Current Date) – Defect Detection Date

2. Defect Density- Defect Density is the number of confirmed defects detected in software/component during a defined period divided by the number of test cases executed

Formula: Defect Density= Number of Defects/Number of test cases executed

3. Test Execution Productivity- It is nothing but the number of test cases that have been created/executed for particular release with is captured in test metrics for estimation purpose

Formula:((Total no of Test cases executed)/ (Effort spent for Test Execution))

4. Test Effectiveness%-(((Total number of application defects based on severity S1+S2+S3&S4) - Total number of application defects rejected by the customer)/ (Total number of application Defects Reported by customer based on severity (S1+S2+S3&S4)+(Total number of application defects based on severity S1+S2+S3&S4))) \*100

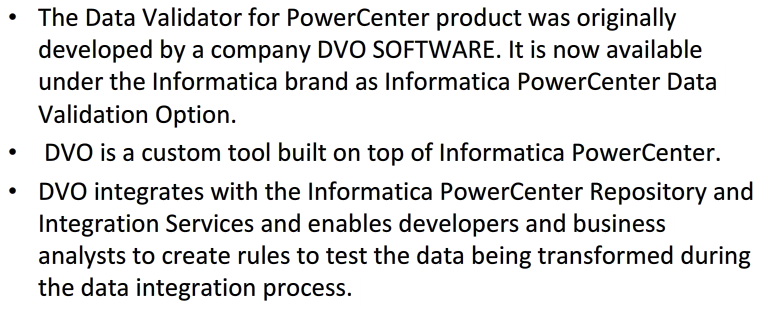
5. Application Defect Leakage % - Valid S1+S2+S3+Rest of the  application Defects Reported by customer/ (Valid S1+S2+S3+Rest of the  Application Defects Found by Cognizant as part of Test Execution + Valid S1+S2+S3+Rest of the  application Defects Reported by the customer)\*100

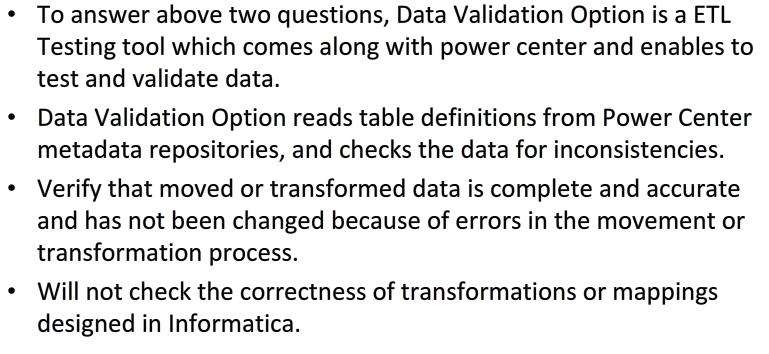
Application

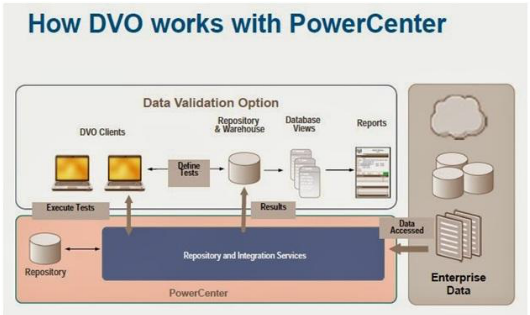
6. Application Defect Density by Size - (Total Number of  S1+S2+S3 + rest of the application defects)/(Size of the work product)

7. Error Discovery Rate - ((Total number of valid application defects based on severity S1+S2+S3&S4) /Number of Test Cases executed (cumulative of the number of cycles executed))

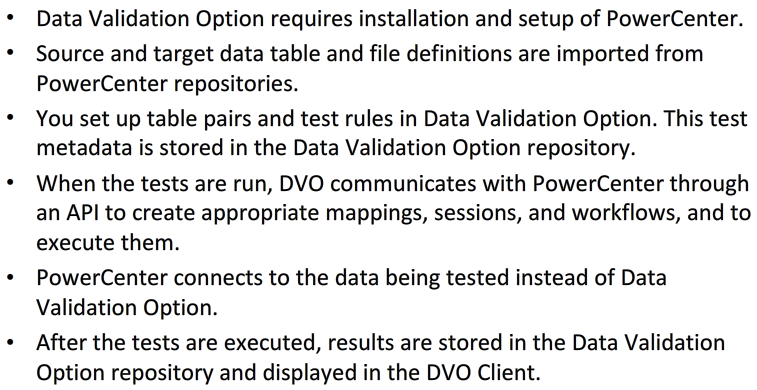
**DVO**

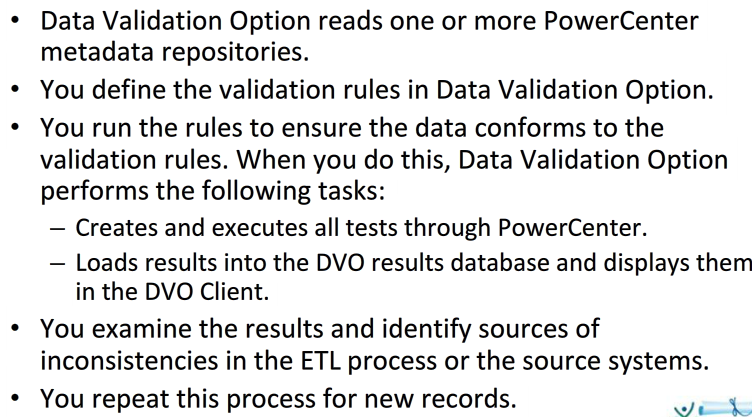




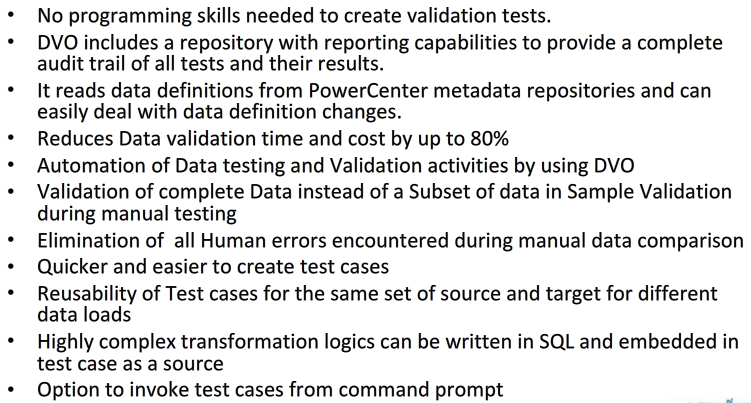


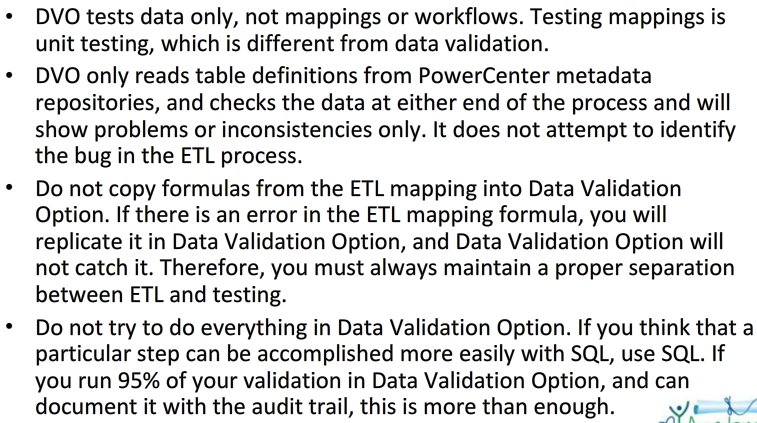
**Continue**

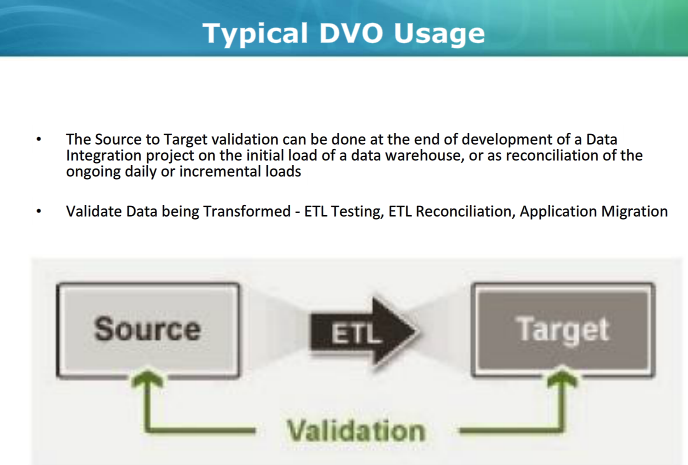


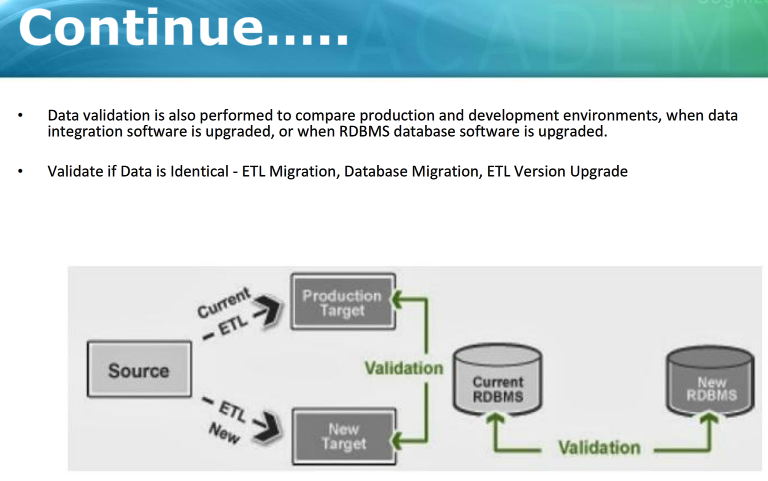


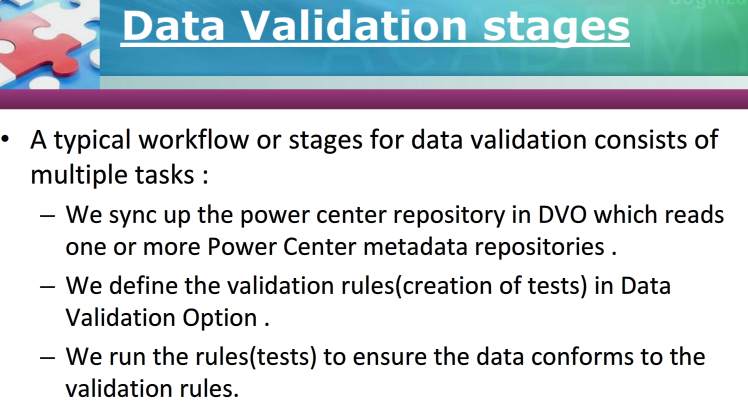
**Key Points:**

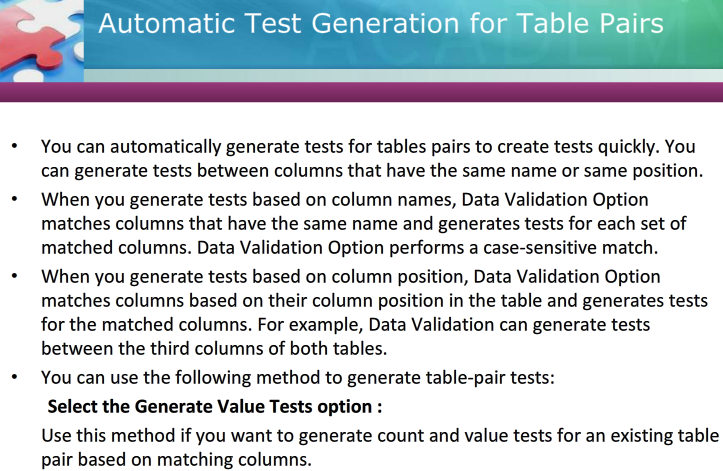


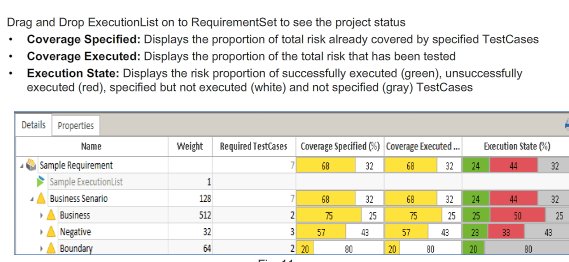












**1. DVO can be used to validate transformation and mappings?**

**2. DVO for Cross-Mapping columns**

**3. Platinum TASQ for full SDLC cycle?**

**4. Available Options in Platinum?**

**5. Indirect links in defect to requirement**

**6. ETL Recovery Testing?**

**7. Conditional sign off**

**8. Sanity testing**

**9. Smoke testing (difference between sanity and Smoke)**

**10. Regression Testing**

**11. UAT Testing**

**12. Advantage of Risk based Testing**

**13. Test data preparation - critical and essential concerns**

**14. Advantage of Agile Testing - Reduces time and cos**

**15. Risk-Based Testing - A good enough testing**

**16. ETL SIT Entry criteria**

**17. History load testing**

**18. Incremental Load testing**

**19. SCD Type 4**

**20. Sanity Testing is Acceptance testing, Smoke is Regression Testing**

**21. Sanity testing checks the major functionality with finer details**

**22. Sanity Testing - Checking the rationality**

**23. Smoke testing - checking the stability**

**24. Platinum Test Case generator**

**25. Risk 1 test case is likely to cause inconvenience to the user**

**26. WSR**

**27. Red Status**

**28. Types of Testing**

**29. Default graph in QC - Bar**

**30. Dynamic update in value during the test execution - "Live Analysis Graph"**

**31. Requirement and functional Defect**

**32. Test configuration is automatically created with the test name in QC**

**33. Pre-populated fields while raising a defect**

**34. How to create a test case in QC**