Planning Analytics Testing Methodology

Software testing Post-upgrade TM1 10.2 to Planning Analytics Local

Post-Upgrade Software Testing

Program Overview & Goals

Program Overview

- Overview of Component, Unit or Functional Test
- Overview & Considerations for Performance Test

Program Goals

- Help plan testing strategies
- Implement testing best practice
- Successful go-live

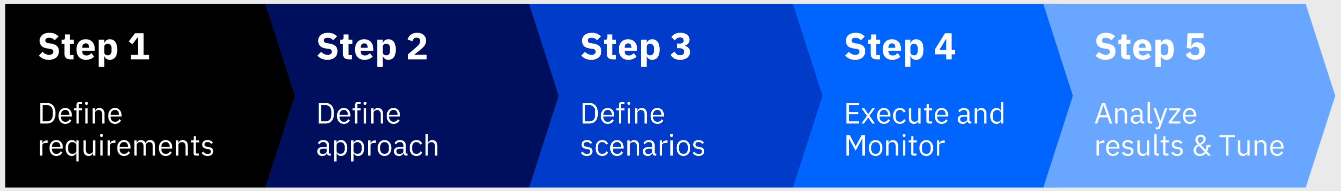
What is Software Testing?

Software testing is a process in which business-critical software is verified for correctness, quality, and performance. Software testing is used to ensure that expected business systems and product features behave correctly as expected.

Benefits of software testing

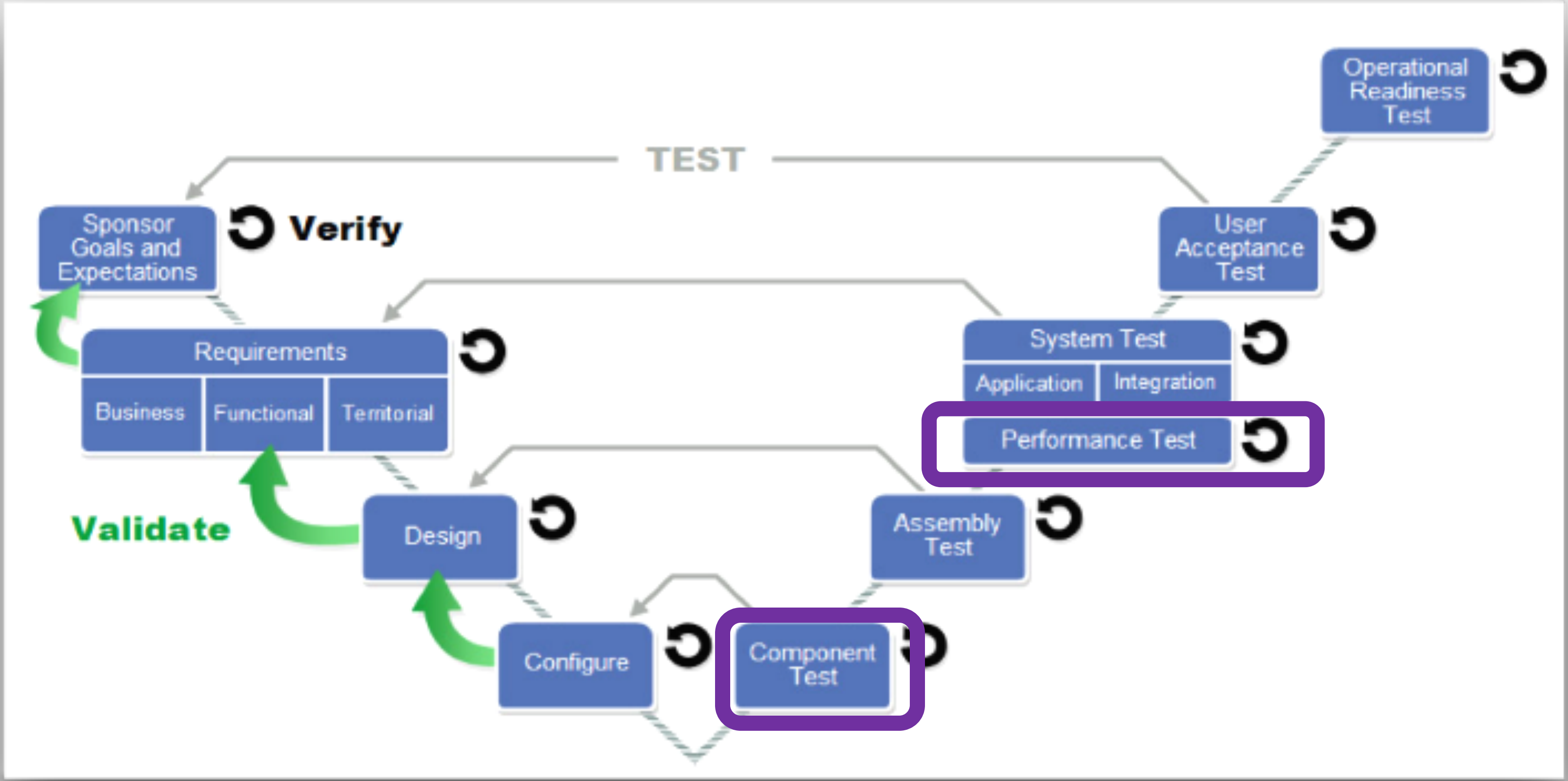
Software testing will help ensure a smooth transition from TM1 to Planning Analytics. Software testing builds stability guarantees into the migration. Testing ensures that a feature is working as expected and users are not encountering defects.

A smooth transition is achieved by specifying a set of test cases that must match the previous version to be considered complete and deliverable. This gives the team a fixed target to work towards enabling more accurate timeline estimates and avoiding the introduction of defects. Once these test cases are in place, the overall maintenance costs are lowered for future upgrades.



Industry Standard V Model

Used for validation and testing purposes. We will focus on Component & Performance test



Levels of Software Testing

Component, Unit or Functional Test

- Minimum required level of testing post-upgrade
- Testing is done at its smallest entity
- Can be executed independently from others
- Ensure no defects in functionality

Performance Test

- Recommended in addition to Component/Unit/Functional tests
- Entire application is tested for functionality and responsiveness
- Can be done manually or with automated tools
- Focus on how your application will perform under expected workload

Component, Unit or Functional Test

What to test?

Minimum Required Testing

- ❑ Test all User Interfaces (UI’s) that are deployed/utilized in the environment
  - ❑ Architect
  - ❑ Performance Modeler
  - ❑ Perspectives
  - ❑ TM1Web
  - ❑ TM1 Applications (previously referred to as TM1 Contributor)
  - ❑ CAFÉ reports in Planning Analytics for Excel
- ❑ Include testing by all locales where Planning Analytics is deployed. If you have end-users with multiple region/language settings, ensure UI testing covers all region/language scenarios
- ❑ Confirm security access in upgraded environment is the same as previous version
- ❑ If Cognos Analytics is integrated, review BI/CA reports to ensure accuracy and with similar performance to previous version
- ❑ Confirm chores run to completion and with similar performance to previous version
- ❑ Confirm Data Source Connection(s) work
- ❑ Review all objects to ensure hardcoded file paths are valid, ex: rules, processes and any reference cubes where file paths are specified

What to test?

- ❑ Dynamic subsets are working and with similar performance to previous version
- ❑ Validate Rule Recalculation is working and with similar performance to previous version
- ❑ Identify business critical end to end processes, for example monthly close process. Execute in parallel in both environments to ensure processes run to completion with similar performance
- ❑ Data Reconciliation
  - ➢ Reconciling is necessary to ensure the validity of data loads and process runs. This is usually a manual task
  - ➢ Possible ways to reconcile:
    - a) Create a view of consolidated elements and check whether these totals match the database totals and AsciiOutPut
    - b) In your model, ensure each dimension has a tree that adds up all the leaf elements. Create a view that pulls these totals. With a tool of your choice, write a query that pulls from your database to compare with this view

Unit Test Scenario example

Validate that the TI process “plan_load_actual_ascii” pulls data from the proper data source and then loads the data into the plan_report cube. Data in the cube should match with what is in the file. Record counts should match.

Unit Test Case

Steps	Testcase	Execution	Expected Result	Additional	Result
1	Validate TI plan_load_actual_ascii is present	Check Architect to ensure the process is available	Yes	N/A	Pass
2	Validate TI plan_load_actual_ascii is present	Check PA Workspace to ensure the process is available	Yes	N/A	Pass
3	Run TI plan_load_actual_ascii	Run TI in Architect	Ran successfully	Architect needs to be installed. TM1 Db is Planning Sample	Pass
4	Run TI plan_load_actual_ascii	Run TI in PA Workspace	Ran successfully	https://PASample.planning-analytics.ibmcloud.com	Pass
5	Review plan_report cube	Check cube for valid data and record counts	Run cube in Architect or PA Workspace	N/A	Pass

Another Unit Test Scenario example

You could focus on MTQ (multi-threaded query) comparing existing version to new upgraded version

Unit Test Case

TM1 Version / # of Threads	Server start-up time / model size	Cube View Load	Flat file load	Reprocessing feeders
10.1 / 1	30:30 / 50gb	2:30	10:50	14:30
10.2 / 4	22:05 / 50gb	1:45	9:05	13:10
Planning Analytics 2.0.6 / 4	18:45 / 50gb	1:30	7:50	13:00
Planning Analytics 2.0.7 / 15	15:30 / 50gb	:25	4:30	10:00

Performance Testing

Importance of Performance Testing

- Recommended in addition to Component/Unit/Functional testing. If not feasible for client, previous environment specs and performance can be used to size upgraded environment.
- Performance Testing is defined as creating a set of processes which will be executed either manually or with assistance of automated tools.
- The aim for running these processes is to test how your application will perform under the expected workload.
    - ➢ Speed – how quickly the application responds
    - ➢ Stability – system is still responsive under full user load
    - ➢ Scalability – application and system able to handle load given
- Performance testing the software will give clients the confidence their system:
    - ➢ Is reliable
    - ➢ Gives required response times under full load
    - ➢ Identify and eliminate performance bottlenecks before they are encountered by end-users

Planning Approach

- **Establish Goals**
  - Assess the impact of having multiple users inputting figures to TM1 cubes
  - Learn the impact of having multiple users inputting figures to TM1 Cubes while executing data update on that same instance
  - Learn the limit of users who can input to TM1 cube
  - At a minimum, bottlenecks and lock contention can generally be identified with as few as 10 simultaneous users. While ideal to simulate expected load, if that isn't feasible, if 10 simultaneous instances of executing X doesn't indicate a problem, it is likely 100/200/300 will show similar results. **Note**: Limited performance testing is much better than no performance testing

- **Ensure basic requirements are known**
  - Testing period: Start and End date
  - Availability of environment to start scripting – if automated tools are being used
  - Availability of full set of data needed at Start Date
  - Exclusive availability of environment - Need to ensure nobody else is on or other scheduled jobs running
  - When will users be available? Where are the users located - Regional or Global?
  - With Automated testing, Virtual Users will be needed. When will these be available?

Planning Approach - Users

- How many users will you be testing with?
- Ensure there is an accurate representation of your regular cycles

User Type	Description
Licensed Users	Total amount of users who can use the application
Active Users	• Number of users who would be logged in at any one time • 10% of Licensed Users
Concurrent Users	• Number of users actively pushing resources • Users truly visible at peak period • 10% of Active Users

- Number of concurrent users could vary from client to client. The above is a guide which typically used as a starting point.

- Some clients like to run their testing period based on 10% of Active users and others like 10% of their Licensed users. Project teams would need to have input as it could depend on their various cycles of reporting, budgeting or forecasting.

Business Process Description

The goal is to identify most used Business Process by your organization. Below are examples.

Process #	Process	Description
P1	Data Entry for Sales	Using input form, input data into Rev_Sales cube
P2	Data Entry for OPEX	Using input form, input data into OPEX cube
P3	Master Data Management	Update ABC-PR dimension
P4	Data Upload via CSV file	Upload data to Rev_Sales cube using csv file

Business Process P1: Data Entry for Sales

The following example are steps which testing would follow

	Steps	Screen shots / expected response time
1	Access Application with: https://www.planning-analytics.ibm.com/	/ 5s
2	Launch Application → Sales Input → Sales_Input_Planning	/ 5s
3	Choose condition for Product, Measure and customer And click Refresh	/ 1 min
4	Input Sales Volume in gray cells	/ 15 min

Business Process P2: Data Entry for OPEX

The following example are steps which testing would follow

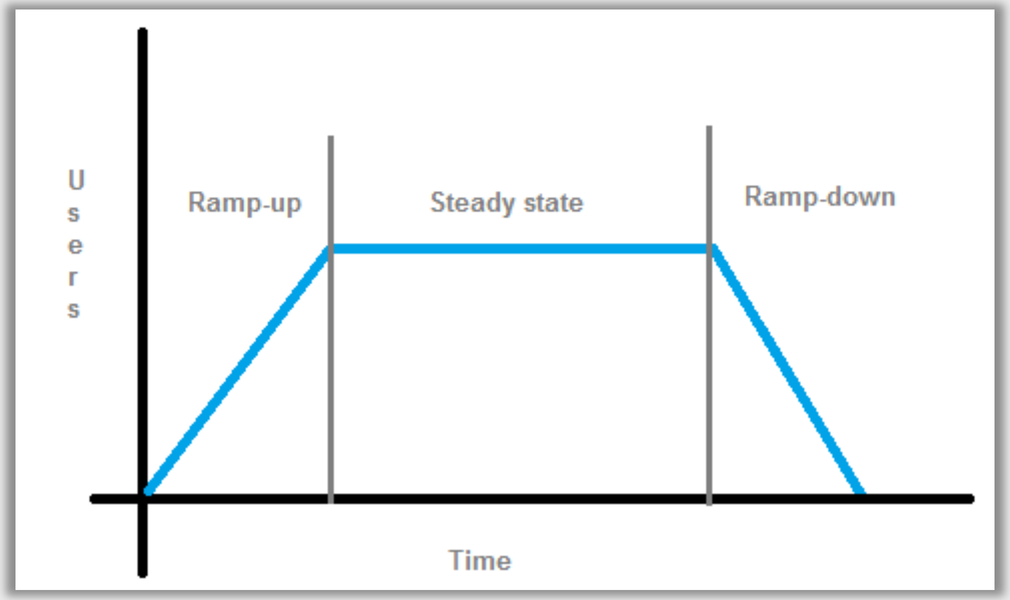
	Steps	Screen shots / expected response time
1	Access Application with: https://www.planning-analytics.ibm.com/	/ 5s
2	Launch Application → Cost Center → Cost_Center_Input_Planning	/ 5s
3	Choose condition for Period, YTD, Cost And click Refresh	/ 1 min
4	Input amounts in gray cells	/ 15 min

Performance Testing Methodology

By definition, the aim of Performance Testing is to determine how a system behaves under a workload in terms of responsiveness and stability. It also helps to identify weaknesses in your application, system resources or even topology.

Typical Flow:

- ➢ Users ramp up gradually to simulate the increase of connections to the application
- ➢ The Steady state gives us performance and stability over time of concurrent actions on the application
- ➢ Once Steady state is over, users ramp down gradually



Manual vs Automated

Manual Testing Pros:

- Fast visual feedback
- Some say less expensive than automation tools
- Human intuition benefits manual element
- No coding like automation tool.

Manual Testing Cons:

- Mistake prone, less reliable
- Unable to record, difficult to reuse
- Time consuming taking up human resources

Automated Testing Pros:

- Has shown to find more defects and bottlenecks compared to human
- Faster and more efficient processes
- Can be recorded
- Flexibility to increase testing coverage

Automated Testing Cons:

- More difficult to get insight on colors, font, or button sizes
- Can be expensive for initial tool and creation of scripts
- Debugging test scripts can be costly

A few common types of Performance Testing

Load Testing

- Understand the ability to perform under anticipated user loads. This would include normal conditions as well as identified peak loads.
- Helps to identify bottlenecks
- Will track how much traffic your software can handle and its response.

Capacity Testing

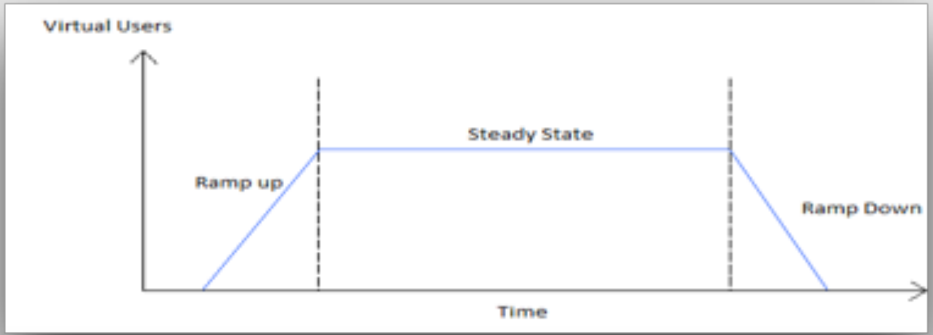
- Evaluating a systems the limit of its anticipated workloads.
- Designed to evaluate an applications behavior when pushed higher than normal peak load conditions.
- Designed to find defects which surface under high load.

Application Stress Testing

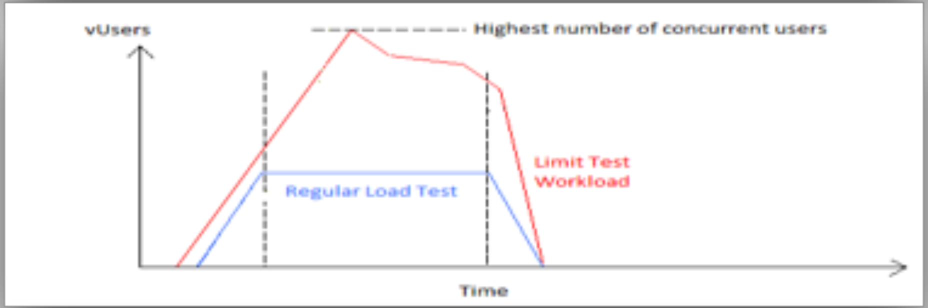
- Focuses more than one transaction on system under stress
- Goal is to uncover defects, data locking, data blocking, network congestion and performance bottlenecks anywhere in the application.
- Common after load testing effort or as a last test phase for capacity planning.

Common Test Approach

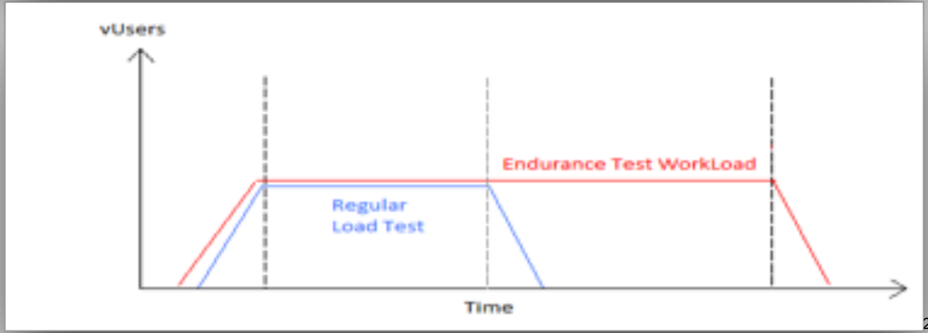
- Load Test: Standard Load Test which reproduces the normal use of the application during an activity peak.



- Capacity Test: Aims to overload the system to analyze it in extreme conditions with an abnormal number of users.



- Endurance Test: Has the same workload as the regular load test but during a longer period of time.



Test Scenario

From what we have learned, we now need to simulate a real world scenario of your application usage

- ➢ Determine the number of concurrent users to be used?
- ➢ Determine which Business Processes will be executed?
- ➢ Determine the frequency of execution?
- ➢ Determine which steady state time you will be using? So how long will the test last
- ➢ Determine Think Time

Example:
Have 25 users execute our Business Process #2
Each doing one iteration every 20 minutes
And execute this for 4 hours

Example of Test Scenarios

- Execute Process #1 (Data Entry Sales), Process #2 (Data Entry for OPEX) and Process #3 (Master Data Management) at the same time.
  - When Process #3 is done, switch to Process #4 (Data Upload via csv file)

Scenario	# of Users	Process per hour
Process #1	25	3
Process #2	20	2
Process #3	1	2
Process #4	1	2

- The idea with this scenario is to start with a small number of users and then increase to the maximum which your organization could have
  - If you determine that your Active Sales users would be 200 Start executing Process #1 with 25 users, then increase to the maximum of 200
  - You can also add an administrator executing Process #4

Scenario	# of Users
Process #1	Start → 25 Gradually increase → 200
Process #4	1

Key Measurements to Track

KPIs	Measurement	Take-aways / Questions
Response KPIs	a) Average response time b) Peak response time c) Error Rate	a) Should be normal expected time b) Time for your longest and most active cycle c) % of problems compared to all requests
Application KPIs	a) CPU Usage b) Memory Usage c) Active threads running d) Active threads waiting	a) Are more CPU being used than anticipated? Not enough being used? MTQ tuning? Is hyperthreading enabled? Can't go faster than processor speed allows b) Healthy? Lots of spikes, normal? Leaks? c) Expected & healthy? d) Expected? Blocked?
Web KPIs	JVM Busy Idle	How healthy is the JVM? If set too low, it will result in slow responses, idle threads, or even crashes.
Host KPIs	CPU usage Memory usage Paging rate Threads I/O disk	

Performance Testing Tools

Open Source

Jmeter

- ➢ Leading tools for Web and Application servers
- ➢ Most popular in the market today
- ➢ https://jmeter.apache.org/

Selenium

- ➢ https://www.seleniumhq.org/

Questions?