QA Artifacts

Software Test Metrics & KPIs

INFO6255

Medi Servat

Software Quality Questions

How long does it take to test?

How is the product performing?

How many bugs are found and how many are still to be found?

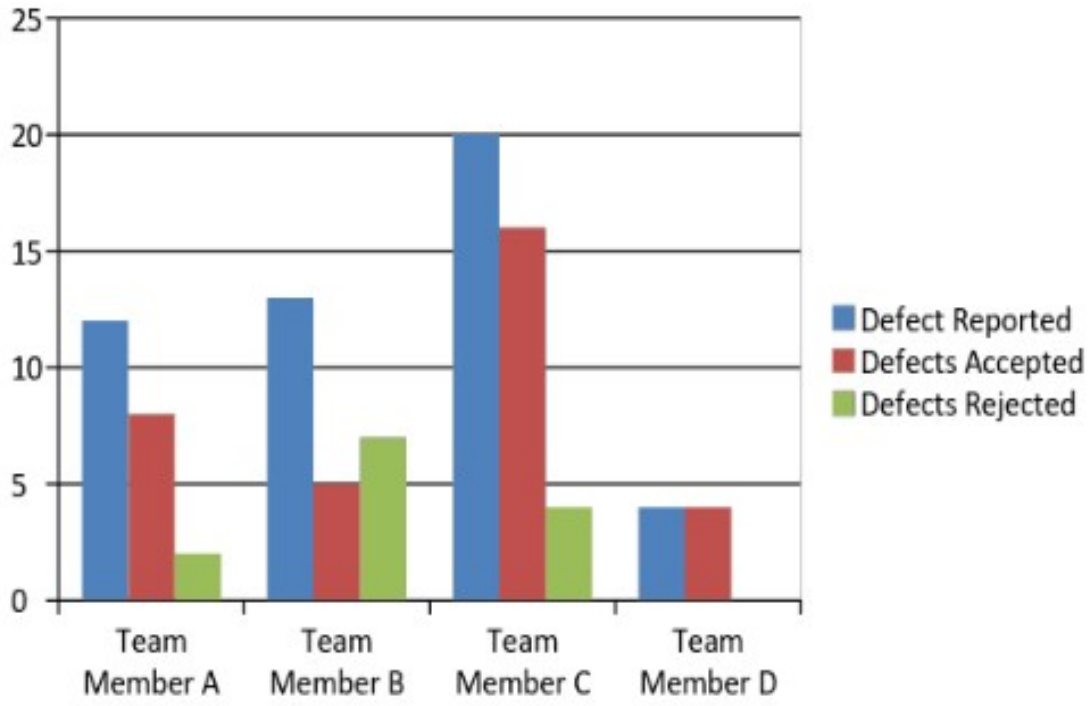
Will the testing be complete on time?

How many resources are needed to finish the testing?

Software Testing Metrics

- Software Testing Metrics can be defined as a **quantitative measure** that helps to make estimates on the software testing projects. It measures the:
  - **Progress**
  - **Quality**
  - **Health**
  - **Test efficiency**
  - And more…

No Testing can be performed without the Software Testing Metrics



https://www.qasymphony.com/blog/64-test-metrics/

What Are Software Test Metrics?

Software test metrics are used to *measure the quality* of the product.
The metrics also improve the *efficiency and effectiveness* of a software testing process.
Without the metrics, how do we measure the quality of work done by the testers?
Examples:
How many test cases have been designed per requirement?
How many test cases are yet to design?
How many test cases are executed?
How many test cases are passed/failed/blocked?
How many test cases are not yet executed?
How many defects are identified & what is the severity of those defects?
How many test cases are failed due to one particular defect? etc.

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Key Performance Indicators

Key performance indicators (KPI) are the important metrics that are calculated by the software testing teams to ensure the project is moving in the right direction and is achieving the target effectively

- **Active Defects:** Identify the status of defects. Allows the team to resolve them.
- **Automated Tests:** Identify and track the number of the automated tests.
- **Covered Requirements:** Track the % of the requirements covered by at least one test.
- **Defects Fixed Per Day:** keep a track of the number of defects fixed on a daily basis.
- **Percentage of Critical & Escaped Defects:** The percentage of critical and escaped defects is an important KPI that needs the attention of software testers
- And more….

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Test Metrics

Based on the metrics, the Test Lead will get the understanding of the following <u>Key Points:</u>

- % of work completed.
- % of work yet to be completed.
- When will the remaining work will be complete?
- Is the project going as per the schedule?
- Is the QA Testing in danger?
- Does the QA manager need to add more resources?

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Test Metrics Types

Two Types of Test Metrics:

- **Base Metrics:** is the raw data collected by Test Analyst during the test case development and execution
  - **i.e. # of test cases executed, # of test cases**).
- **Calculated Metrics:** are derived from the data collected in base metrics. Calculated metrics is usually followed by the test manager for test reporting purpose
  - **% Complete, % Test Coverage**

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com; Guru99.com

Data From a Test Analyst Performing the Testing

11/13/2020

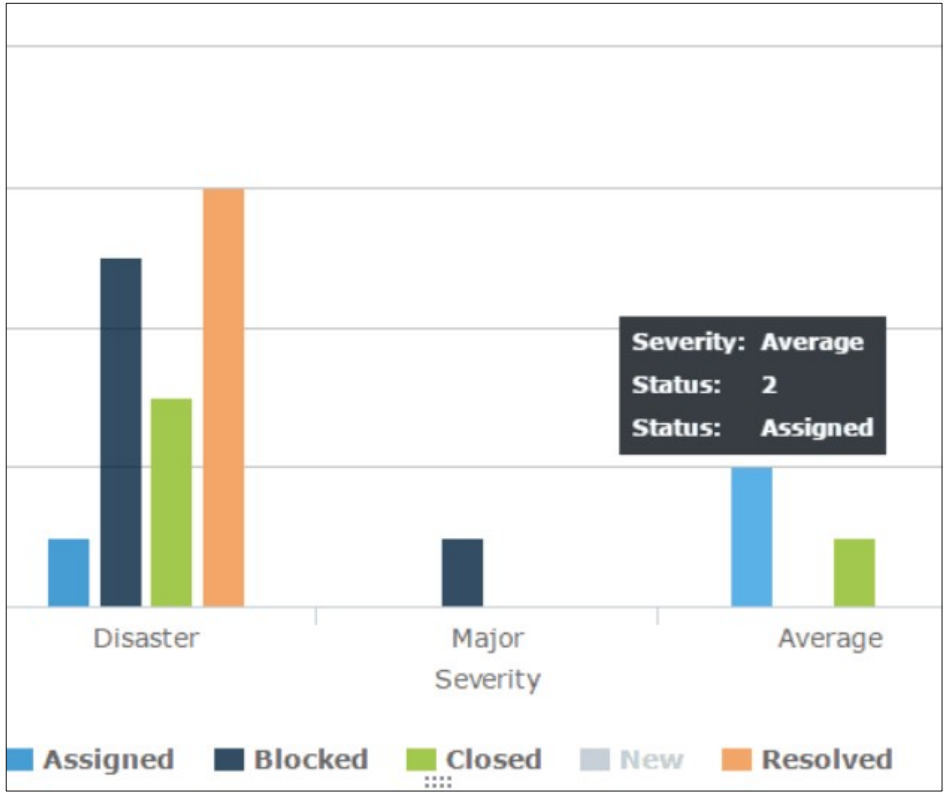
S.No.	Testing Metric	Data retrieved during test case development & execution
1	No. of Requirements	5
2	Avg. No. of Test cases written per Requirement	20
3	Total no. of Test cases written for all requirements	100
4	Total no. of Test cases Executed	65
5	No. of Test cases Passed	30
6	No. of Test cases Failed	26
7	No. of Test cases Blocked	9
8	No. of Test cases un executed	35
9	Total No. of Defects identified	30
10	Critical Defects count	6
11	High Defects Count	10
12	Medium Defects Count	6
13	Low Defects Count	8

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Base and Calculated QA Metrics

- **Examples of Base Metrics:**
  - No of Test Cases Executed
  - No of Test Cases Passed/Failed/Blocked
  - Not of Defects discovered

- **Examples of Calculated Metrics:**
  - % of Test Cases Executed
  - % of Test Cases Passed/Failed/Blocked
  - % of Test cases Remaining
  - % of Defects Fixed



https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Test Coverage Metrics

Process Metrics + Product Metrics Test Coverage Metrics

- **Process Metrics:** Are used to measure and enhance processes of Software Development, Maintenance and Testing.
  - **It is used in the process of test preparation and test execution phase of SDLC.**

- **Product Metrics:** It measures the end-product quality built by the development and QA.
  - It is used in the process of defect analysis phase of SDLC.

Process Metrics

- **Test Case Preparation Productivity:** It is used to calculate the number of Test Cases prepared and the effort spent for the preparation of Test Cases :

Test Case Prep Productivity = (No of Test Case) / (Effort spent for Test Case Preparation)

EXAMPLE:

- No. of Test cases = 240
- Effort spent for Test case preparation (in hours) = 10
- Test Case preparation productivity = 240/10 = 24 test cases/hour

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Process Metrics

- **Test Design Coverage:** It helps to measure the percentage of test case coverage against the number of requirements

Test Design Coverage = ((Total number of requirements mapped to test cases) / (Total number of requirements))*100

EXAMPLE:

- Total number of requirements: 100
- Total number of requirements mapped to test cases: 98
- Test Design Coverage = (98/100)*100 = 98%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Process Metrics

- **Test Execution Productivity:** it determines the number of Test Cases that can be executed per hour

Test Execution Productivity = (No of Test cases executed)/ (Effort spent for execution of test cases)

EXAMPLE:

- No of Test cases executed = 180
- Effort spent for execution of test cases (Hours) = 10
- Test Execution Productivity = 180/10 = 18 test cases/hour

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Process Metrics

- **Test Execution Coverage:** It is to measure the number of test cases executed against the number of test cases planned.

Test Execution Coverage = (Total no. of test cases executed / Total no. of test cases planned to execute)*100

EXAMPLE:

- Total no. of test cases planned to execute = 240
- Total no. of test cases executed = 160
- Test Execution Coverage = (160/240)*100 = 75%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Process Metrics

- **Test Cases Passed:** It is to measure the percentage number of test cases passed

Test Cases Passed = (Total no. of test cases passed) / (Total no. of test cases executed) * 100

EXAMPLE:

- Test Cases Pass = (80/90)*100 = 88.8 = 89%

- **Test Cases Failed:** It is to measure the percentage no. of test cases failed

Test Cases Failed = (Total no. of test cases failed) / (Total no. of test cases executed) * 100

EXAMPLE:

- Test Cases Failed= (10/90)*100 = 11.1 = 11%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Process Metrics

- **Test Cases Blocked:** It is to measure the percentage no. of test cases blocked

Test Cases Blocked = (Total no. of test cases blocked) / (Total no. of test cases executed) * 100

EXAMPLE

- Test Cases Blocked = (5/90)*100 = 5.5 = 6%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Process Metrics

- **Effort variance** (EV) calculates **variance** of actual **effort** versus planned **effort**.

Effort variance = [(Actual effort- Planned Effort)/Planned effort] * 100

- The **effort variance** may be greater than expected. **For example, we estimated 100 hours but actual work took 110 hours**.

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Process Metrics

- **Test effectiveness** metrics usually show a percentage value of the difference between the number of defects found by the **test** team, and the overall defects found for the software.

Test efficiency = (total number of defects found in Unit + Integration + System) / (total number of defects found in Unit + Integration + System + User Acceptance testing)

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Test Coverage Metrics

Process Metrics + Product Metrics Test Coverage Metrics

- **Process Metrics:** Are used to measure and enhance processes of Software Development, Maintenance and Testing.
  - It is used in the process of test preparation and test execution phase of SDLC.

- **Product Metrics:** It measures the end-product quality built by the development and QA.
  - **It is used in the process of defect analysis phase of SDLC**.

Product Metrics

- **Error Discovery Rate:** It is to determine the effectiveness of the test cases.

(Total number of defects found /Total no. of test cases executed)*100

EXAMPLE:

- Total no. of test cases executed = 240
- Total number of defects found = 60
- Error Discovery Rate = (60/240)*100 = 25%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Product Metrics

- **Defect Fix Rate:** It helps to know the quality of a build in terms of defect fixing.

Defect Fix Rate = (Total no of Defects reported as fixed - Total no. of defects reopened) / (Total no of Defects reported as fixed + Total no. of new Bugs due to fix)*100

EXAMPLE:

- Total no of defects reported as fixed = 10
- Total no. of defects reopened = 2
- Total no. of new Bugs due to fix = 1
- Defect Fix Rate = ((10 – 2)/(10 + 1))*100 = (8/11)100 = 72.7 = 73%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Product Metrics

- **Defect Density:** It is defined as the ratio of defects to requirements.

  Defect density determines the stability of the application.

Defect Density = Total no. of defects identified / Size (requirements)

EXAMPLE:

- Total no. of defects identified = 80
- Actual Size= 10
- Defect Density = 80/10 = 8

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Software Test Metrics – **Product Metrics**

- **Defect Leakage:** It is used to review the efficiency of the testing process before UAT. **How many defects are missed/slipped during the QA testing.**

Defect Leakage = ((Total no. of defects found in UAT)/(Total no. of defects found before UAT)) * 100

EXAMPLE:

- No. of defects found in UAT = 20
- No. of Defects found before UAT = 120
- Defect Leakage = (20 /120) * 100 = 16.6 = 17%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Product Metrics

- **Defect Removal Efficiency:** It allows us to compare the overall (defects found pre and post-delivery) defect removal efficiency

Defect Removal Efficiency = ((Total no. of defects found pre-delivery) /((Total no. of defects found pre-delivery)+ (Total no. of defects found post-delivery)))* 100

EXAMPLE

- Total no. of defects found pre-delivery = 80
- Total no. of defects found post-delivery = 10
- Defect Removal Efficiency = ((80) / ((80) + (10)))*100 = (80/90)*100 = 88.8 = 89%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Product Metrics

- **Defects by Priority:** is used to identify the no. of defects identified based on the Severity / Priority of the defect which is used to decide the quality of the software

% Critical Defects = No. of Critical Defects identified / Total no. of Defects identified * 100

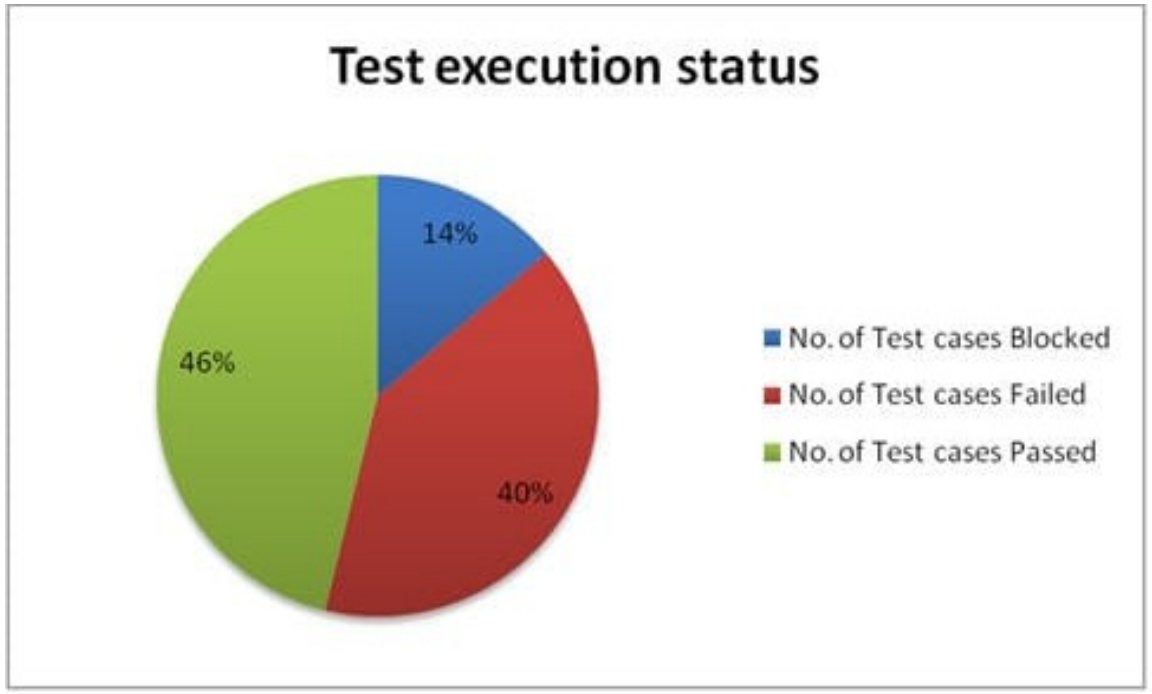
% High Defects = No. of High Defects identified / Total no. of Defects identified * 100

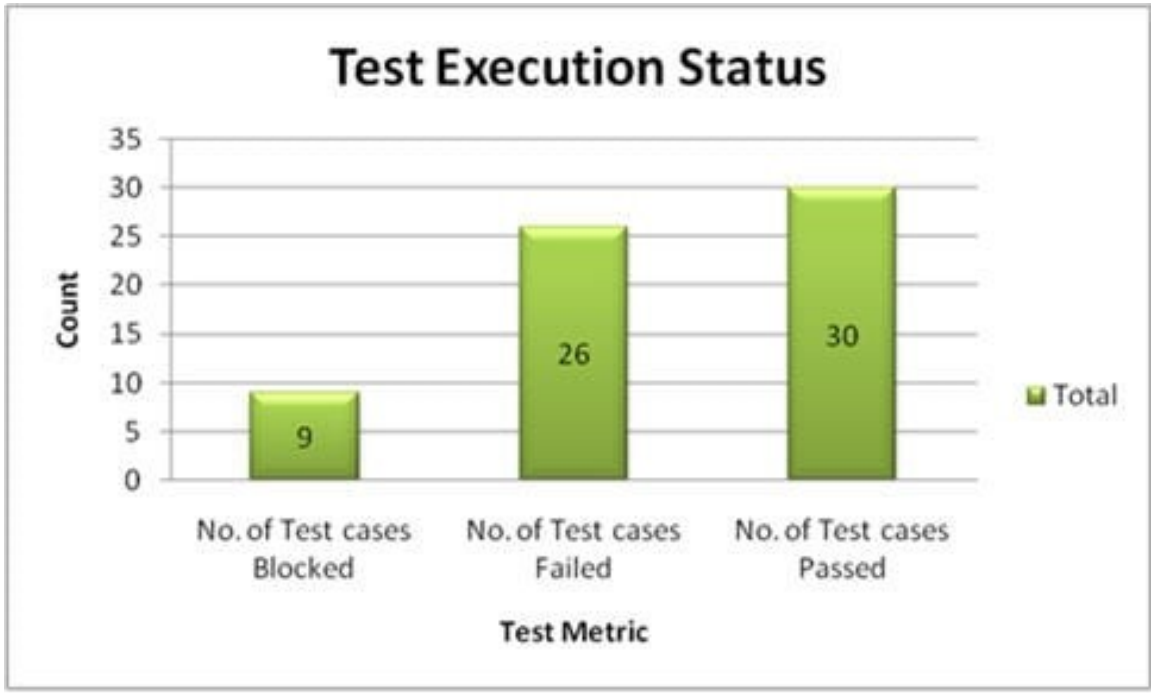
EXAMPLE:

Percentage of Critical Defects = (6/ 30) * 100 = 20%

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Test Execution Examples





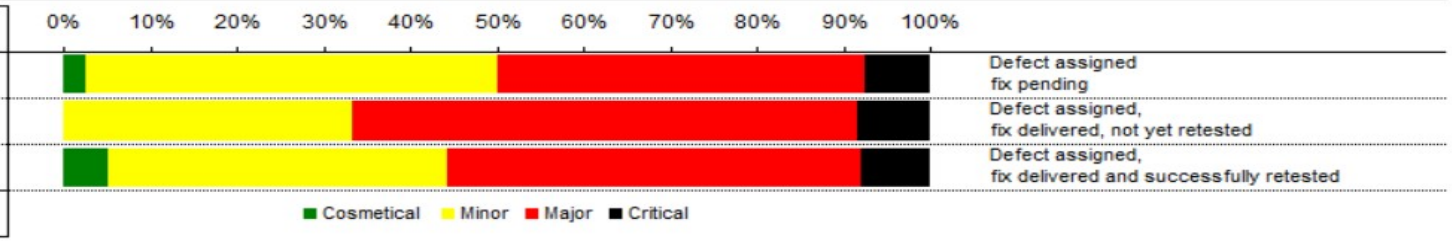
https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Software Development - Defect Dashboard

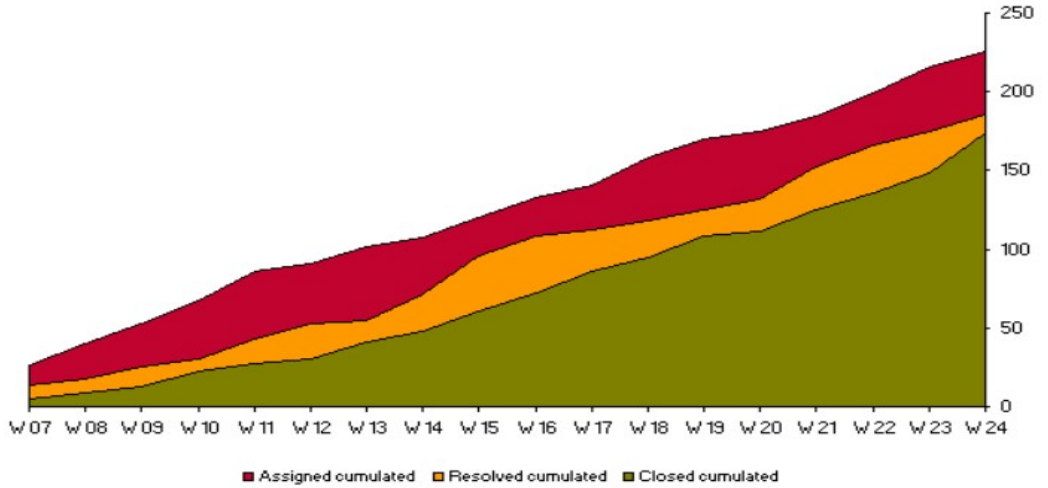
from Mo, 06/08/09 to So, 06/14/09 Week 24

Defects actual week by type and severity

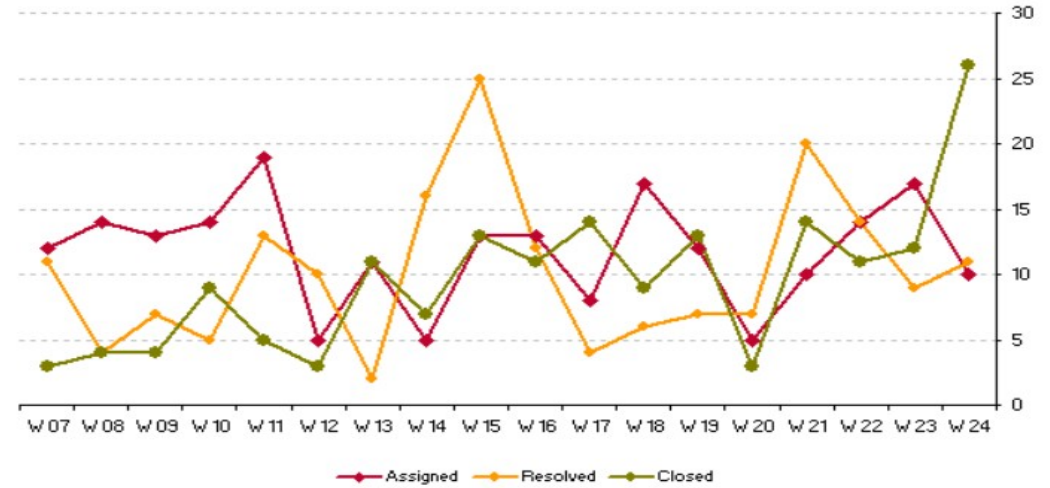
	Cosmetical	Minor	Major	Critical	Total
assigned	1	19	17	3	**40**
resolved	0	4	7	1	**12**
closed	9	68	83	14	**174**
Total	**10**	**91**	**107**	**18**	**226**



Assigned, resolved and closed defects cumulated over the last 18 weeks



New defects assigned, resolved and closed last 18 weeks - week on week



Average resolution / conclusion times in calendar days

	Cosmetical	Minor	Major	Critical	Total
Assigned to resolved	10,8	16,6	18,2	9,1	16,5
Resolved to closed	6,7	14,1	11,0	5,2	11,5
Assigned to closed	17,4	30,7	30,0	14,3	28,4

Average resolution / conclusion times in calendar days last 18 weeks

More QA Metrics

- **Rework Effort Ratio =** (Actual rework efforts spent in that phase/ total actual efforts spent in that phase) X 100
- **Requirement Creep =** (Total number of requirements added/No of initial requirements)X100
- **Schedule Variance =** (Actual efforts – estimated efforts) / Estimated Efforts) X 100
- **Cost of finding a defect in testing =** (Total effort spent on testing/ defects found in testing)
- **Schedule slippage =** (Actual end date – Estimated end date) / (Planned End Date – Planned Start Date) X 100
- **Fixed Defects Percentage** = (Defects Fixed/Defects Reported) X 100
- **Accepted Defects Percentage** = (Defects Accepted as Valid by Dev Team /Total Defects Reported) X 100
- **Defects Deferred Percentage** = (Defects deferred for future releases /Total Defects Reported) X 100

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

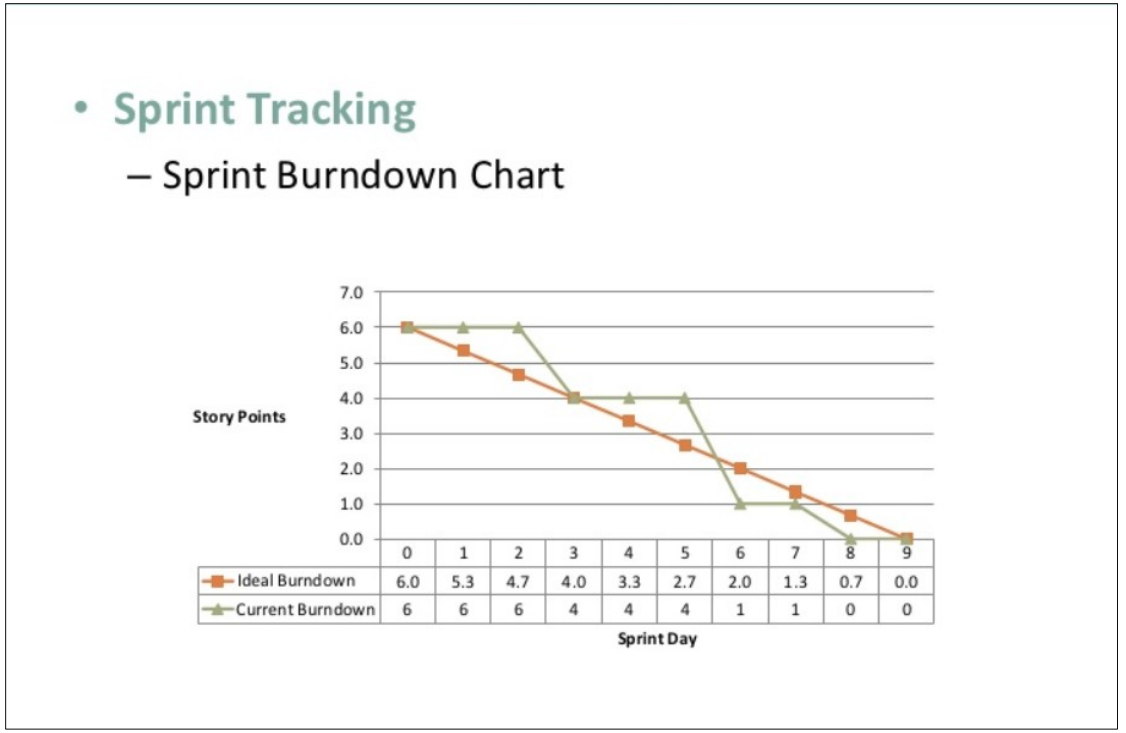
More QA Metrics…

- **Critical Defects Percentage** = (Critical Defects / Total Defects Reported) X 100
- **Average time for a development team to repair defects** = (Total time taken for bugfixes/Number of bugs)
- **Number of tests run per time period** = Number of tests run/Total time
- **Test design efficiency** = Number of tests designed /Total time
- **Test review efficiency** = Number of tests reviewed /Total time
- **Requirement stability index (RSI) =** is a metric used to organize, control, and track changes to the originally specified requirements for a new system project or product.
- **Bug find rate or Number of defects per test hour** = Total number of defects/Total number of test hours

https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Agile Test Metrics

- **Sprint burndown** – How much work is remaining in the current iteration, and how much testing remains to be done.



https://www.sealights.io/test-metrics/

4 Types of Burndown Charts

- **Product burndown chart:** A graph which shows how many Product Backlog Items (User Stories) implemented/not implemented.

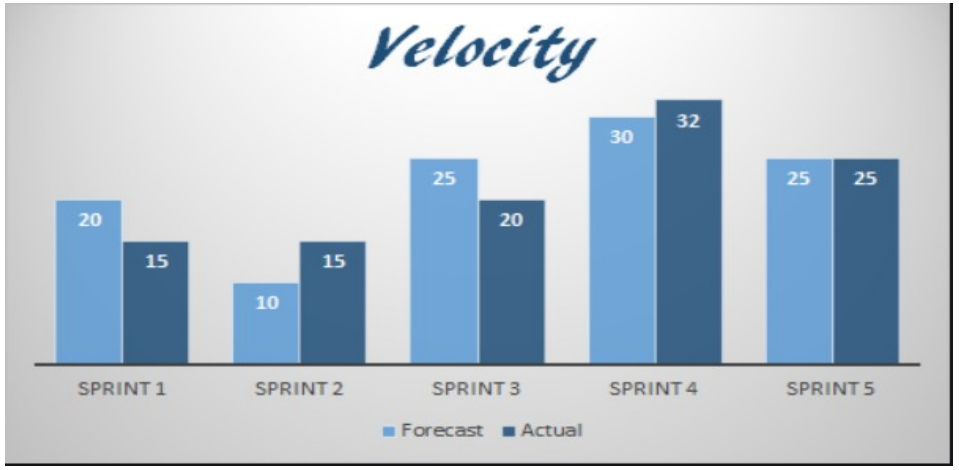
- **Sprint burndown chart:** A graph which shows how much work is remaining in the current iteration, and how much testing remains to be done.

- **Release burndown chart:** A graph which shows list of releases still pending, which Scrum Team have planned.

- **Defect burndown chart:** A graph which shows how many defects identified and fixed.

Agile Test Metrics…

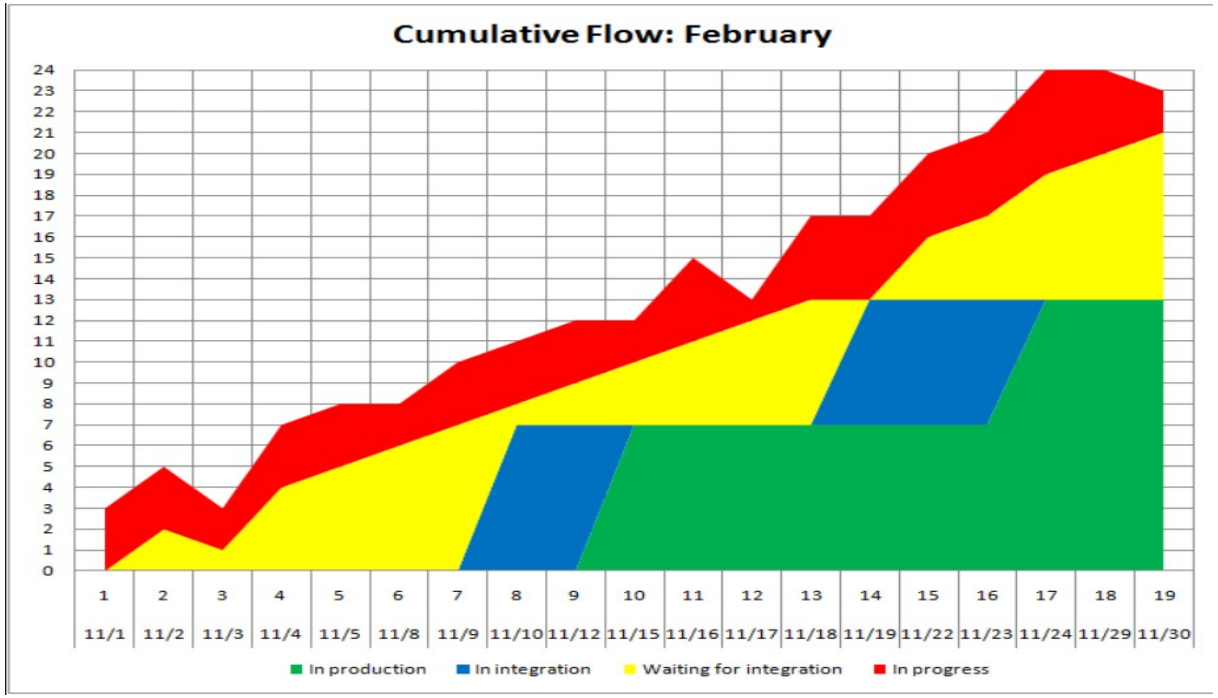
- **Number of working tested features / running tested features** – The more features or agile “stories” are consistently added to the software and fully tested, the healthier the project.
- **Velocity** – The **speed** at which the development team is completing new features.
  - It predicts how much work an agile software development team can successfully complete within a sprint and how much time will it need to finish a project.



https://www.sealights.io/test-metrics/

Agile Test Metrics…

- **Cumulative flow** – Helps visualize ***bottlenecks in the agile*** process.
  - In particular, helps teams visualize if testing resources are adequate and if testing is slowing down the development cycle.



https://www.sealights.io/test-metrics/

Test Report

Test Cycle System Test

EXECUTED	PASSED			130	
	FAILED			0	
	(Total) TESTS EXECUTED (PASSED + FAILED)				**130**
PENDING					0
IN PROGRESS					0
BLOCKED					0
(Sub-Total) TEST PLANNED (PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)					**130**

Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	*High*	
Edit Customer	Check Customer can be edited	100%	100%	0	*High*	
New Account	Check New account is added	100%	100%	0	*High*	
Edit Account	Check Account is edit	100%	100%	0	*High*	
Delete Account	Verify Account is delete	100%	100%	0	*High*	
Delete customer	Verify Customer is Deleted	100%	100%	0	*High*	
Mini Statement	Verify Ministatement is generated	100%	100%	0	*High*	
Customized Statement	Check Customized Statement is generated	100%	100%	0	*High*	

References

- https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/
- NeilPatel.com
- Guru99.com
- https://www.softwaretestingmaterial.com/test-metrics/#ProcessMetrics