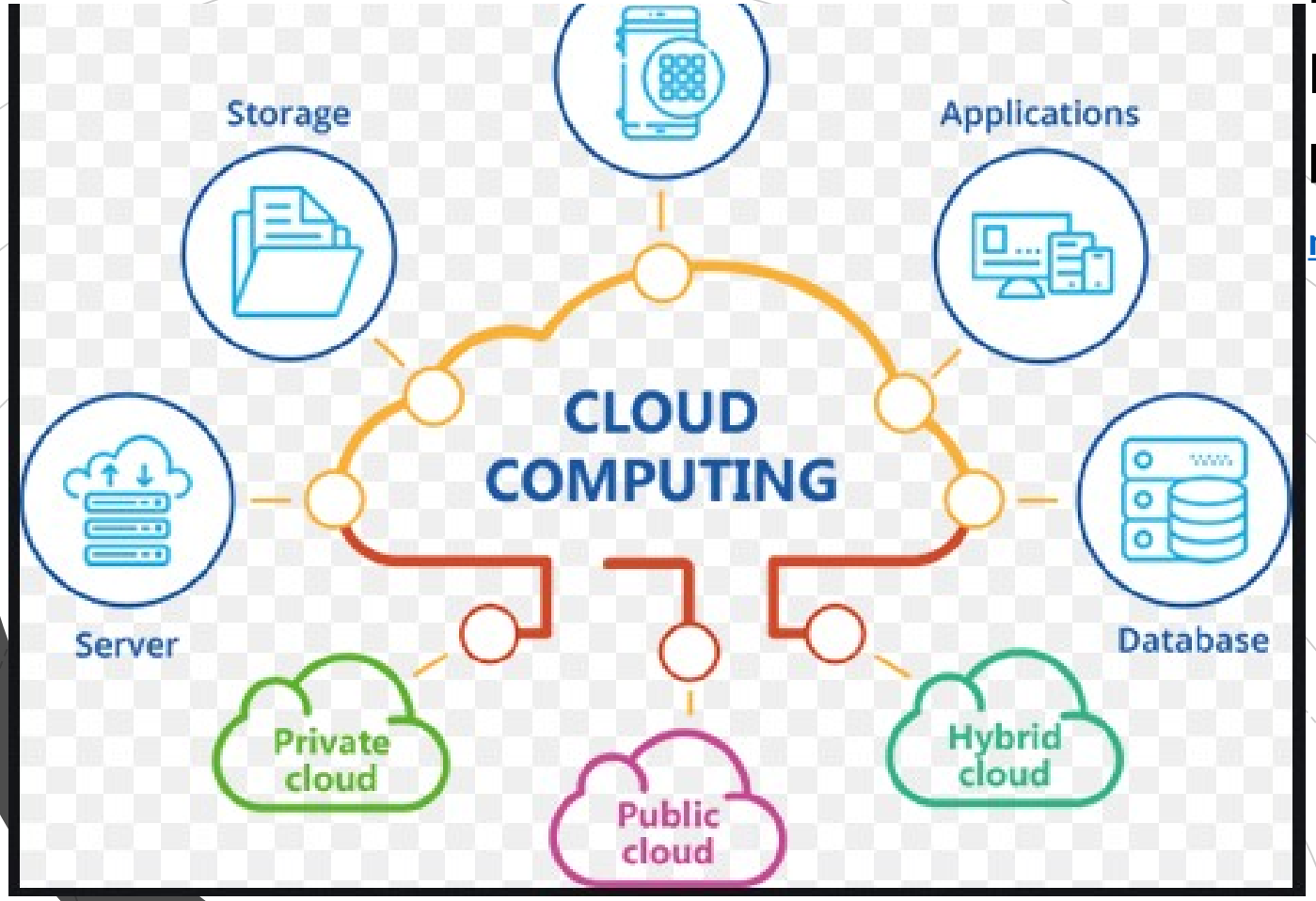


Northeastern University

INFO 6255

Fall 2020

m.Servattalab@northeastern.edu

What Is Cloud Computing

- **Cloud Computing** means storing and accessing data and programs over the Internet instead of your computer's hard drive. **The cloud is just a metaphor for the Internet.**

- It is the act of using an interconnected network of Internet-hosted remote servers to manage, store and process data.

- It is used for everything from banking to data sharing and, because it allows multiple computers on a local network to access data, it has become a popular form of technology among companies of all sizes.

Uses of Cloud Computing

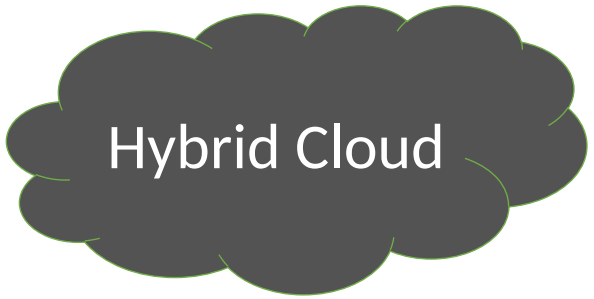
A Cloud allows an online service to send email, edit documents, watch movies or TV, listen to music, play games, or store pictures and other files.

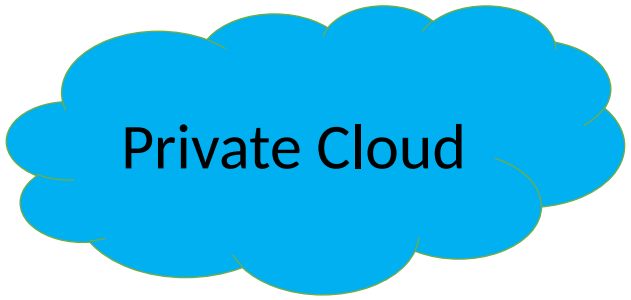
- **Create cloud-native applications**
  - Quickly build, deploy, and scale applications—web, mobile, and API. Take advantage of cloud-native technologies and approaches, such as containers, Kubernetes, microservices architecture, API-driven communication, and DevOps.
- **Store, back up, and recover data**
  - Protect your data more cost-efficiently—and at massive scale—by transferring your data over the Internet to an offsite cloud storage system that's accessible from any location and any device.
- **Test and build applications**
  - Reduce application development cost and time by using cloud infrastructures that can easily be scaled up or down.
- **Analyze data**
  - Unify your data across teams, divisions, and locations in the cloud. Then use cloud services, such as machine learning and artificial intelligence, to uncover insights for more informed decisions.

Cloud Deployment Models

A **cloud deployment model** represents a specific type of cloud environment, which are primarily distinguished by size, access and ownership.

There are 3 common cloud deployment models





Public Cloud

•**Public Cloud:** are owned and operated by a third-party cloud service providers which deliver their computing resources, like servers and storage, over the internet.

examples

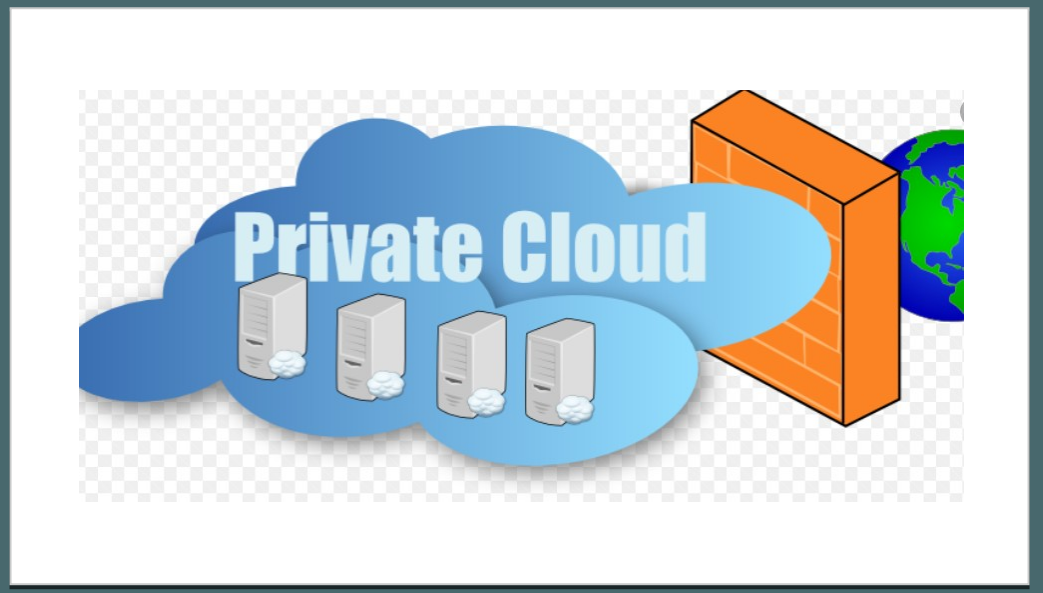
- Microsoft Azure, Amazon Web Services, Salesforce are examples of public cloud.
- All hardware, software, and other supporting infrastructure is owned and managed by the **Cloud provider**.
- Anyone can access these services and manage your account using a web browser.

Private Cloud

Private Cloud: Cloud computing resources are used exclusively by a single business or organization.

- It can be physically located on the company's on-site data center. Some companies also pay third-party service providers to host their private cloud.

- In **private cloud** the services and infrastructure are maintained on a private network.



Hybrid Cloud

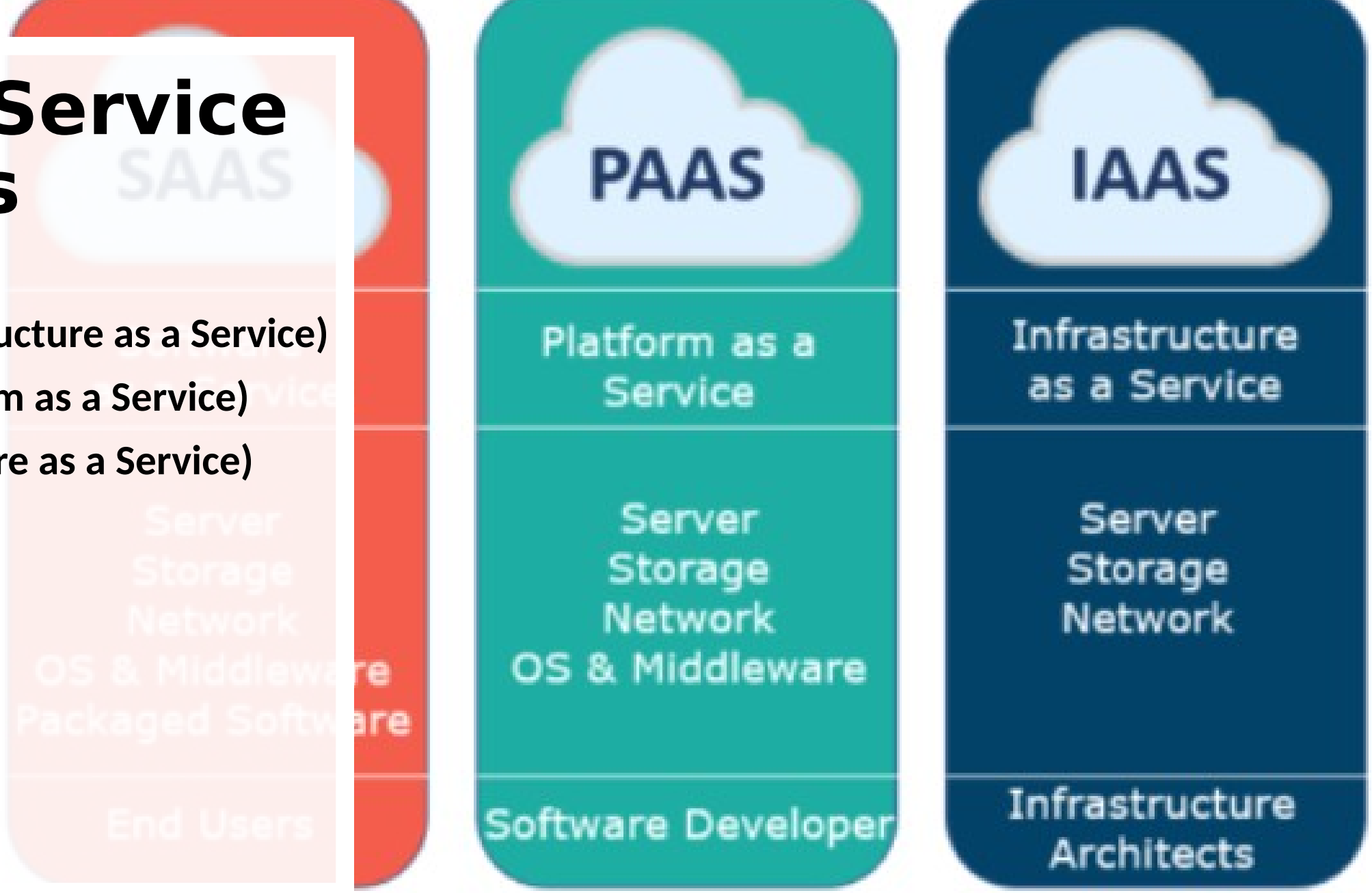
Hybrid Cloud: A hybrid cloud is a combination of private cloud and the public cloud services allowing data and applications to move between private and public clouds.

优点

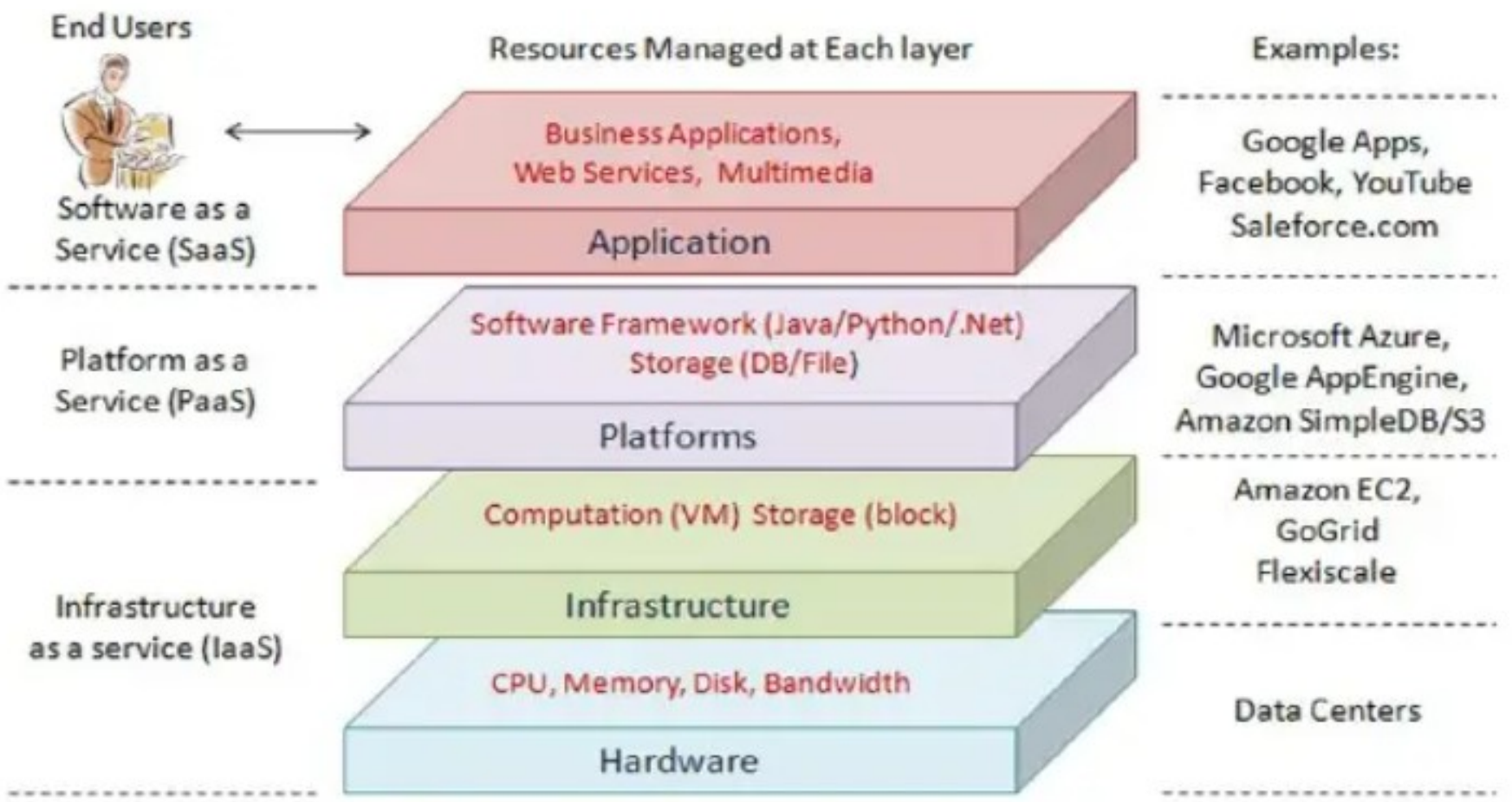
This model gives businesses greater flexibility and more deployment options.

Cloud Service Models

- IaaS (Infrastructure as a Service)
- PaaS (Platform as a Service)
- SaaS (Software as a Service)



Cloud Layer Architecture



Cloud Service Models

IaaS (Infrastructure as a Service): Vendor provides users access to computing resources such as **servers, storage and networking**. Organizations use their own platforms and applications within a service provider's infrastructure.

Key Features:

- **Infrastructure is scalable** depending on processing and storage needed
- Teams can quickly set up and dismantle test and development environments, bringing new applications to market faster.
- It is a **quick and economical way to scale up** dev-test environments up and down.
- Saves businesses the cost of purchasing and maintaining their own hardware. Users pay for IaaS on demand.
- Organizations avoid the capital outlay for storage and complexity of storage management, which typically requires a skilled staff to manage data and meet legal and compliance requirements.

Cloud Service Models...

PaaS (Platform as a Service): Provides users a cloud environment in which they can develop, manage and deliver applications.

It is an IaaS Service offering, plus middleware, development tools, business intelligence (BI) services and database management systems.

Key Features:

- PaaS is designed to support the complete web application lifecycle: building, testing, deploying, managing, and updating.
- Providers manage security, operating systems, server software and backups.
- Facilitates collaborative work even if the teams are working remotely.

Cloud Service Models...

SaaS (Software as a Service): Provides users with access to **a vendor's cloud-based** software. Users do not install applications on their local devices, instead cloud providers host and manage the software application and underlying infrastructure, and handle maintenance like upgrades and system patching.

Key Features:

- SaaS makes sophisticated enterprise applications, such as ERP and CRM, affordable for organizations that lack the resources to buy, deploy, and manage the required infrastructure and software themselves.
- You also save money because the SaaS service automatically scales up and down according to the level of usage.
- Data is secure in the cloud; equipment failure does not result in loss of data.

Cloud Based Testing

- **Cloud-based testing** can be applied for testing cloud, web, and installed applications.
- Providers of cloud testing services and tools offer test environments that can be configured according to application's requirements.
- Cloud testing has given rise to **Testing as a Service (TaaS)**, which allows organizations to outsource their testing efforts.
- **TaaS** can be used for overall software testing as well as for conducting specialized types of testing such as performance, security, or functional testing.
- Cloud testing focuses on the core components like:
  - **Application:** It covers testing of functions, end-to-end business workflows, data security, browser compatibility, etc.
  - **Network**: It includes testing various **network bandwidths, protocols** and successful transfer of data through networks.
  - **Infrastructure**: It covers **disaster recovery test, backups, secure connection, and storage policies.** The infrastructure needs to be validated for regulatory compliances

Benefits of Cloud-Based testing

18页有两句话总结

In contrast to traditional software testing, cloud-based testing has several unique advantages:

1. **Scalability**: Cloud computing allows testers to increase or decrease computing resources according to their needs.
   - This is very useful in cases when the client frequently **changes their business requirements**.

2. **Cost-Cutting:** In cloud computing, you pay only for those resources that you use.
   - This means that there's no need to invest in **expensive equipment and spend money** maintaining and upgrading it.
   - There can be all the **software and hardware you might need at your disposal** while only paying for it when you actually use it.

3. **Easily Customizable:** By using cloud-based tools and services, testers can easily emulate an end-user-centric environment with minimum cost and time.
   - The test team can perform various types of testing in any combination of device environments.

Benefits of Cloud-Based Testing

4. **Ensure Comprehensive Testing:** For comprehensive testing, the test team needs to run an application on all possible devices that support different platforms, OS, and browsers.
   - Cloud-based testing provides you with all these devices and configurations, eliminating the need to purchase all of them.

5. **Faster Testing:** Cloud-based testing tools ensure automated testing, which greatly reduces the time to market for software**.**
   - This is achieved by the ease of building testing infrastructure, increased collaboration within the test team.

6. **Constant Availability:** Software testing in the cloud is available to testers at any time and anywhere.
   - So testers can speed up software deployment and testing.

Types of Testing in the Cloud

The Cloud Testing is segmented into four main categories:

1. **Testing of the whole cloud**: The cloud is viewed as a whole entity and based on its features testing is carried out.
   - Cloud and SaaS vendors, as well as end users, are interested in carrying out this type of testing
2. **Testing within a cloud**: By checking each of its internal features, testing is carried out.
   - Only cloud vendors can perform this type of testing
3. **Testing across cloud**: Testing is carried out on different types of cloud-like private, public and hybrid clouds.
4. **SaaS testing in cloud**: Functional and non-functional testing is carried out on the basis of application requirements.

Testing Type	Description
Performance Testing	While cloud solutions should be scalable on demand, this type of testing ensures that the application performs correctly with various numbers of users.
Security Testing	This type of testing is necessary for ensuring that data is stored and transmitted safely. Security mechanisms of applications are tested according to three criteria: **effectiveness, accuracy, and performance**. The most popular tools for testing security in the cloud are **Nessus, Wireshark and Nmap**.
Functional Testing	Functional software testing checks all the features and functions of software and its interaction with hardware. For conducting functional testing, testers can use such tools as **TimeShiftX, Rapise and Sauce Labs**
Load & Stress Testing	During load testing, testers measure software response time while the system is subjected to increasing load. It's also necessary to check how the application will work under **excessive stress**. If you want to measure the application response delay after deploying it in the cloud, then you can conduct latency testing.
Interoperability & Compatibility Testing	**It is carried out to validate** the compatibility requirements of the application under test system. It checks browser compatibility in a cloud environment. It identifies the Defects that might arise while connecting to a cloud. It is carried out to verify if application works across a different platform of cloud.

Top 10 Cloud-based Software Testing tools

Testing in the cloud brings with it benefits of **easy availability, high scalability and low cost**.

It allows for web and mobile testing in different environments and multiple machines without building your own infrastructure.

Here are some of the hugely popular cloud-based software testing tools:

1. **SOASTA CloudTest**
2. **LoadStorm**
3. **BlazeMeter**
4. **Nessus**
5. **App Thwack**
6. **Jenkins Dev@Cloud**
7. **Xamarin Test Cloud**
8. **TestLink**
9. **Test collab**
10. **Test Sigma**

Cloud Testing vs. Conventional Testing

Testing Parameters	Conventional Testing	Cloud Testing
Primary Testing Objective	Check interoperability, compatibility, usability. • Verifies the quality of system function and performance based on the given specification.	Verifies the quality of performance and functions of SaaS, Clouds, and applications by leveraging a cloud environment.
Testing Costs	Costing remains high due to hardware and software requirements.	Only have to pay for operational charges • Pay only for what is being used.
Functional Testing	Validating functions (unit and system) as well as its features.	Testing end-to-end application function on SaaS or Cloud.
Testing Environment	A pre-fixed and configured test environment in a test lab.	An open public test environment with diverse computing resources.
Integration Testing	Component, architecture, and function-based testing.	SaaS-based Integration Testing.
Security Testing	Testing security features based on process, server and privacy.	Testing security features based on cloud, SaaS and real time tests in vendor's cloud.
Performance and Scalability Testing	Performed in a fixed test environment.	Apply both real time and virtual online test data.

References

- https://www.apriorit.com/dev-blog/548-cloud-based-testing
- https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#cloud-computing-models
- https://www.investopedia.com/terms/c/cloud-computing.asp

Testing Blockchain Transactions & Cryptocurrency

Northeastern University

INFO 6255

Fall 2020

m.Servattalab@northeastern.edu

What is Blockchain?

- Blockchain is a decentralized data source.

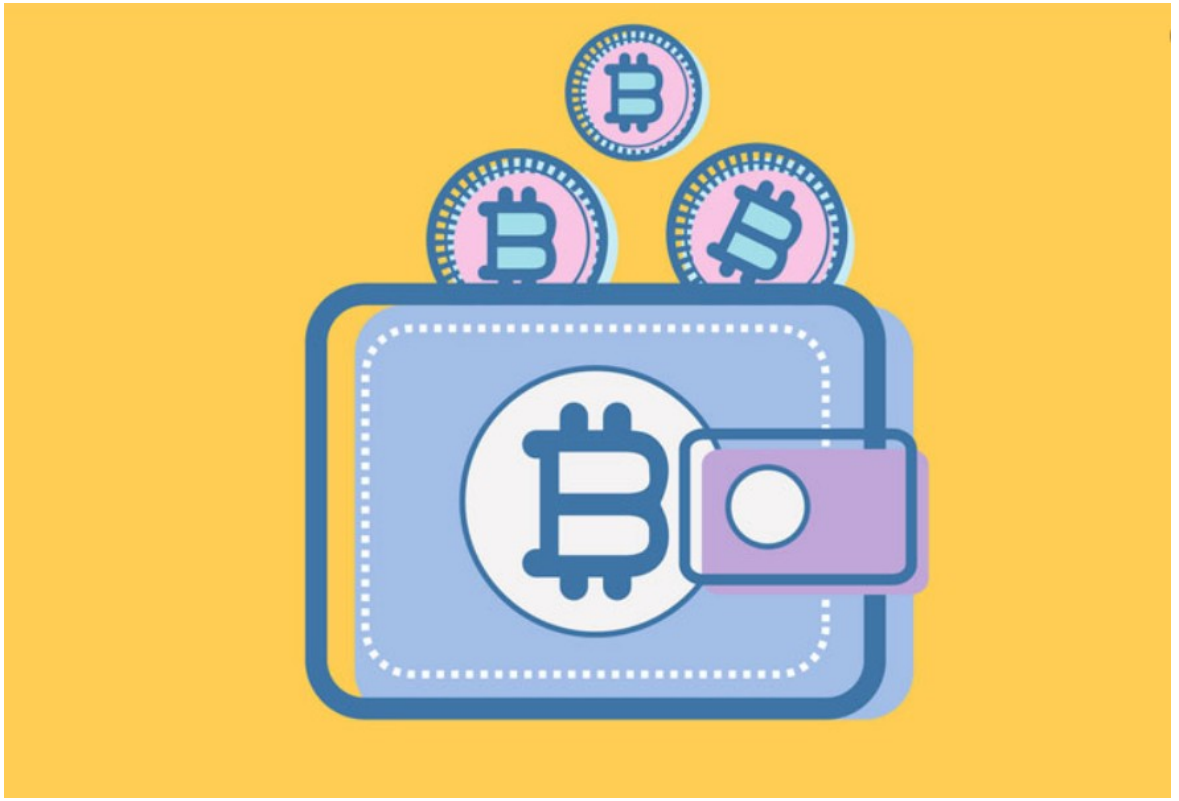
- Blockchains rely on data being verified by a number of different sources controlled by different entities, instead of a central repository.

- Information is confirmed by multiple disparate sources, and each piece of information builds on the previous one.

https://blog.gurock.com/testing-blockchain-cryptocurrency

What is Blockchain?

- One example: **Bitcoin**
  - The blockchain is a ledger of transactions detailing how currency has moved from one address to another, with multiple versions being updated simultaneously and continuously by multiple parties.



https://blog.gurock.com/testing-blockchain-cryptocurrency

Testing Digital Currency Constraints

- When testing **digital currency** transactions on a blockchain, there are some important constraints:
  - You cannot manipulate or change data to create certain outcomes
  - There are limited options for testing without real assets
  - It’s difficult to control the timing of your transactions

Testing Digital urrency Constraints…

- Unable to manipulate data for testing to create different situations or create different errors.

- When you're testing on a public blockchain, you have no control over where crucial transaction data is sourced or stored

Testing Digital Currency Constraints…

- There are two separate bitcoin blockchains to consider for testing:
  - The **mainnet chain:** It is the real bitcoin chain. Transactions here utilize the genuine digital currency
  - The **testnet chain:** It's transactions function in the same way, except these are test currency and are never supposed to have any value.
    - **Testnet** can be a great way to test transactions.
    - You can send as much as you want, and there's no real cost to your organization.
    - You still need to get the funds. One of the easiest ways to acquire **testnet** funds is to use a Faucet.
    - **Faucets** are websites set up by other Bitcoin developers with funds that can be shared

https://blog.gurock.com/testing-blockchain-cryptocurrency

Testing Digital Currency Constraints…

- When you're setting up tests for blockchain transactions, you may need to **be prepared to wait**.
  - Block confirmations take time.
- One way to try to get your transactions **picked up** by the network more quickly is to send higher amounts of money
  - You'll be charged a **higher fee** for sending a larger amount
- Another situation where timing can be a problem is when your wallet syncs directly with the **public blockchain**
  - Tests that involve a full sync require advanced planning.

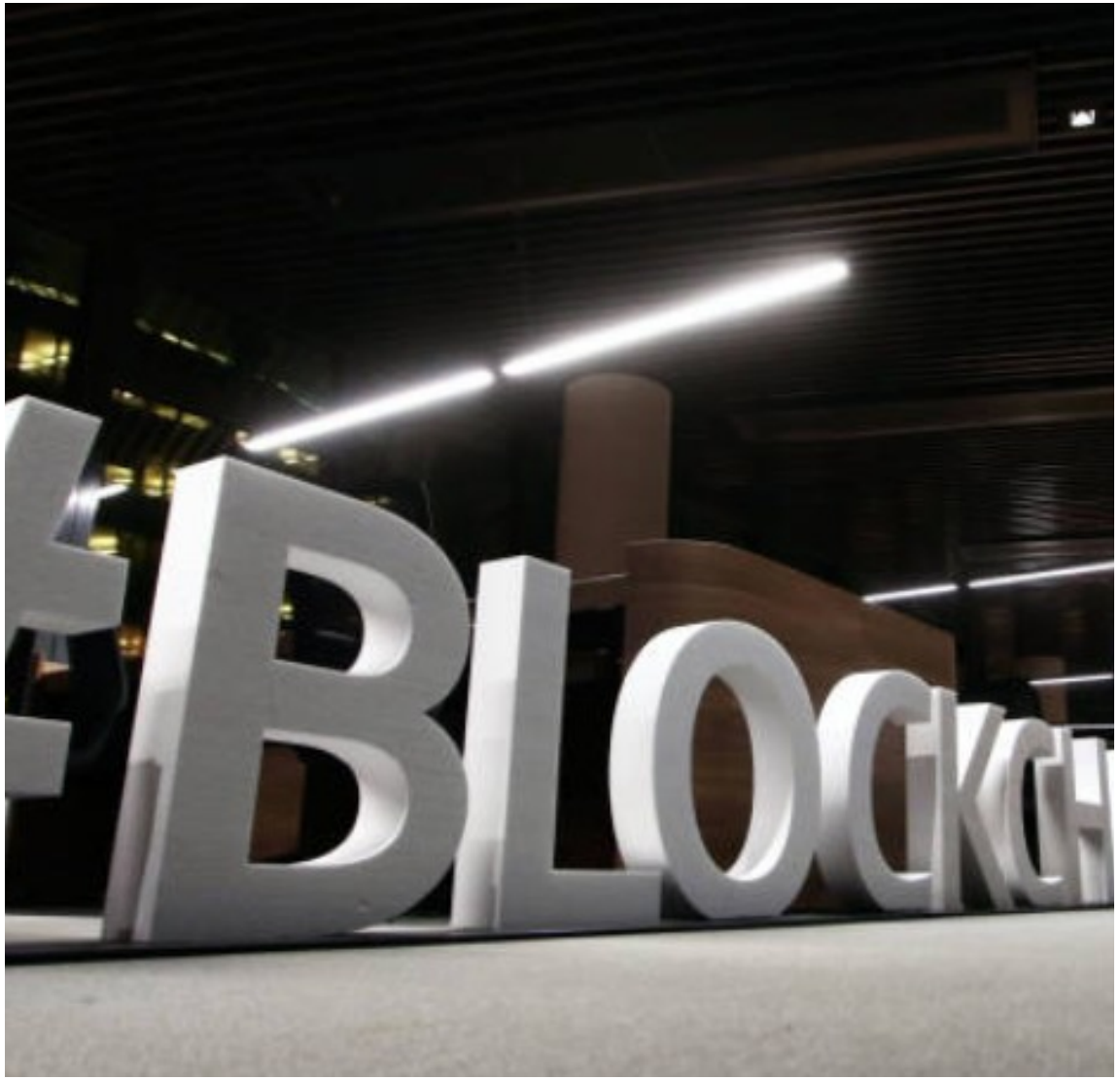
Crypto Wallet Transaction Verification

Block Explorer: is a website that allows you to search for transactions on the blockchain.

- You can search and quickly find a number of reliable options; many are set up for a variety of blockchains.

CryptoCurrenty Addresses:

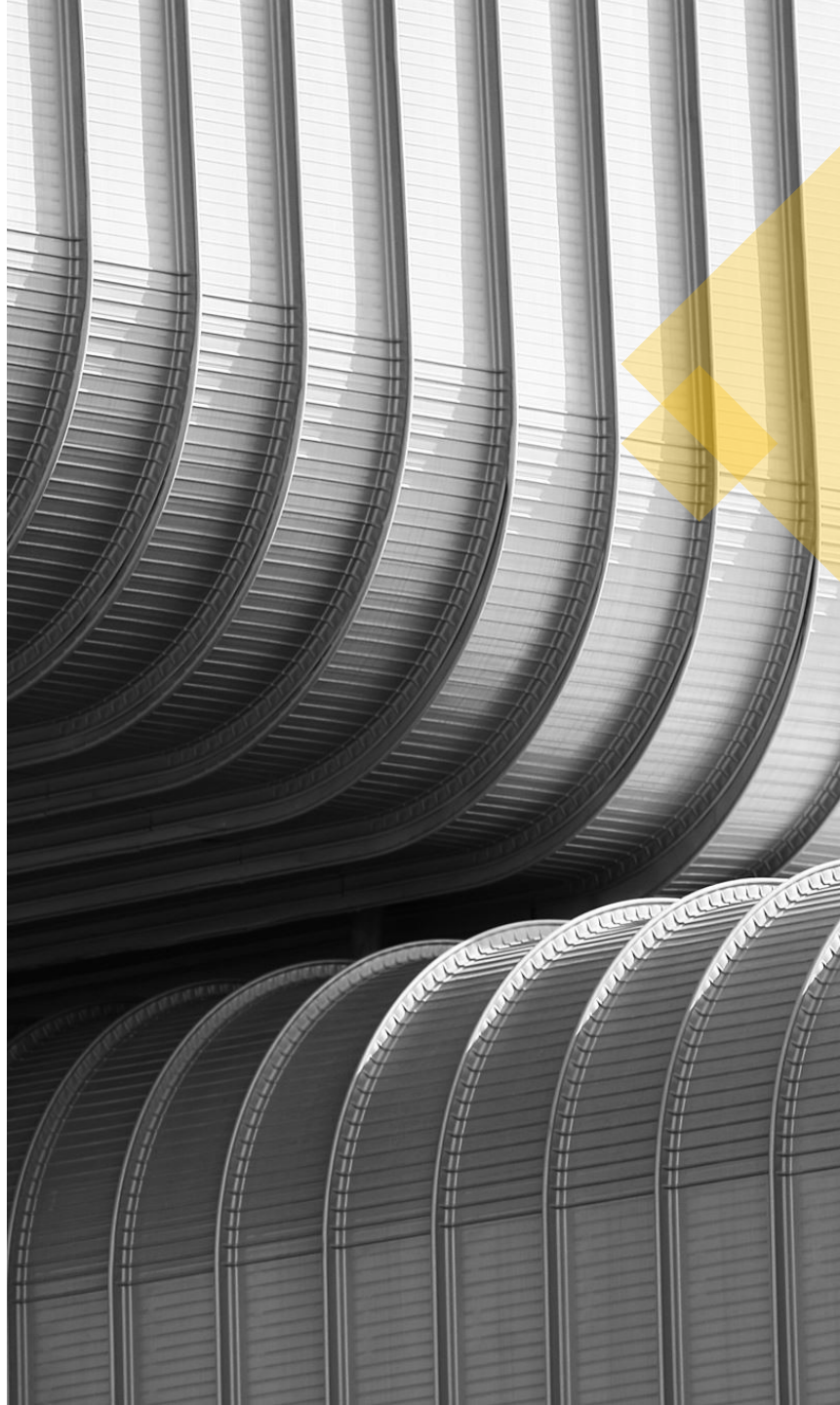
- Crypto investors have the ability to transact with anonymity.
- Your crypto address is created by the wallet and tied to a private key that is stored on your device.
- Your address is available publicly but not tied to any of your personal identifying information.



Crypto wallet transaction Verification…

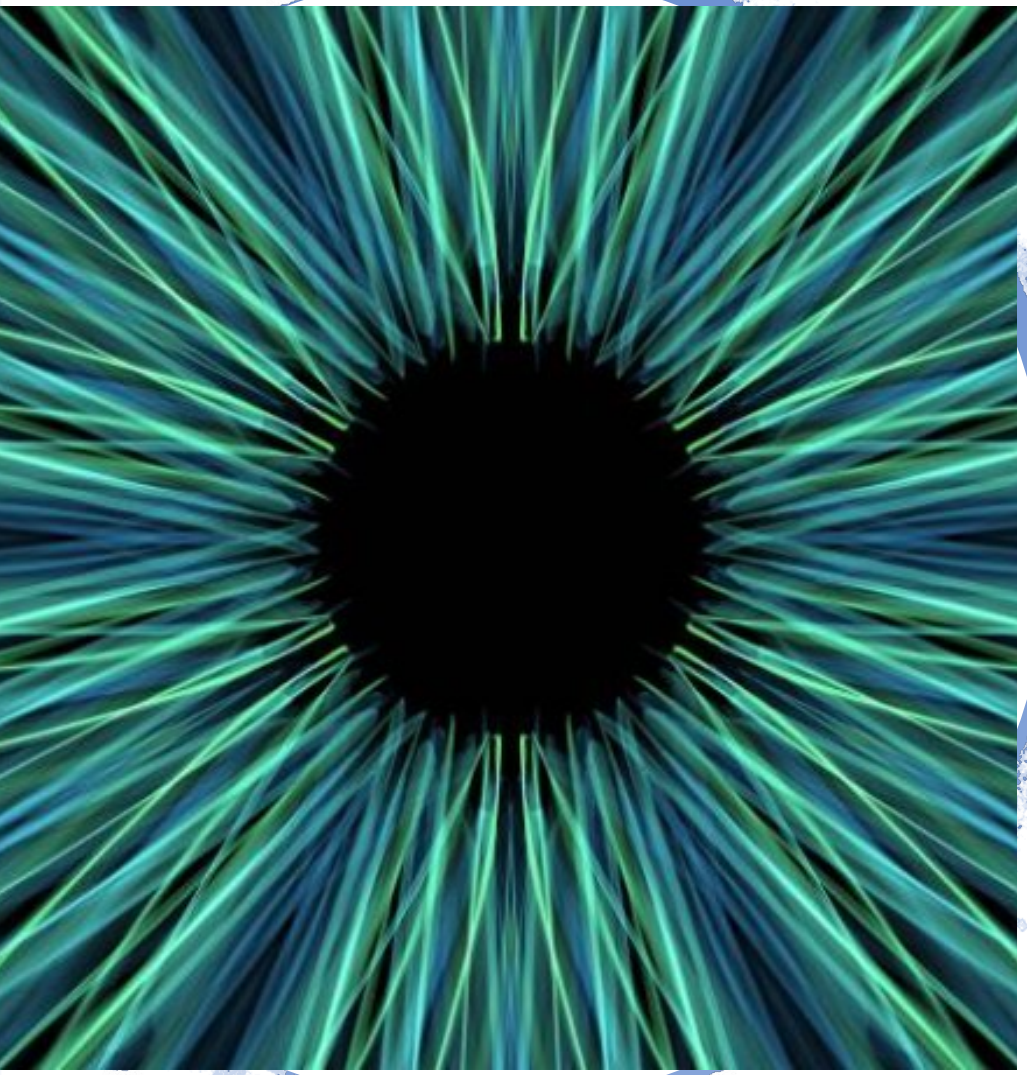
To ensure that the transaction has been registered properly on the blockchain with the correct value:

- **Transaction Accuracy:**
  - Verify transactions in three locations:
    - The sending wallet
    - The receiving wallet
    - On the blockchain
- **Block confirmations**
  - Blockchain transactions need to be confirmed before they can be finalized.
  - Some wallets may have rules about when to display a transaction as pending or completed, based on the number of block confirmations.



Crypto wallet transaction Verification…

- **Memos or tags**
  - **Memos or tags**, are important for certain tokens.
  - **For example,** Ripple’s digital currency, XRP, uses a destination tag with transactions.
  - This identifier helps ensure that transactions on the XRP blockchain are routed correctly.
- **Fees or gas**
  - Each digital currency has its own rules and fee structure.
  - Fees on the **Ethereum network**. The Ethereum blockchain has a number of currencies that can be exchanged, called ERC20 tokens.
  - Unlike Bitcoin, whose fees are paid in bitcoin, any fees on the Ethereum blockchain are paid in **Ethereum**, instead of the token you are sending.



Summary

- **Blockchain** presents a seemingly never-ending place to learn and explore.
- On the surface, some aspects of this technology are easy to understand, but things get complex very quickly.
- You can learn as much as you can about all the rules of any given chain.
- It can be overwhelming at first, but there are a wealth of online sources

Reference

- https://blog.gurock.com/testing-blockchain-cryptocurrency/?utm_source=newsletter&utm_medium=email&utm_campaign=blog-nl-al-em-2020-09-30

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?

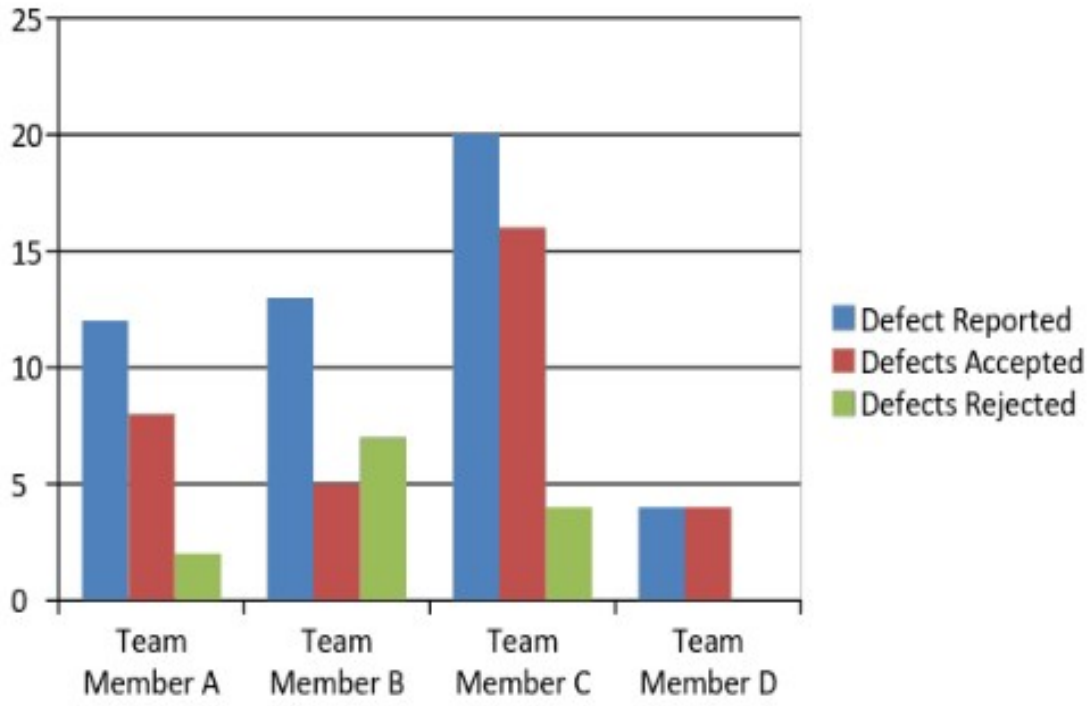
Will the testing be complete on time?

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…

下一页也有定义



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

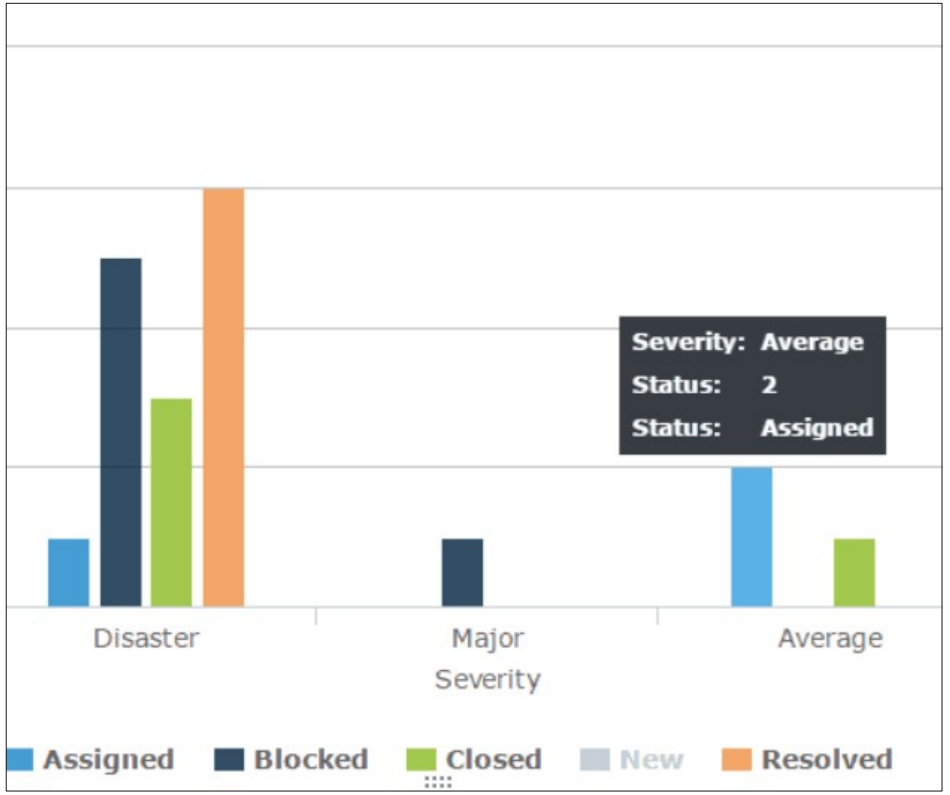
S.No.	Testing Metric	Data retrieved during test case development & execution
1	No. of Requirements	5
2	Avg. No. of Test cases written per Requirement	20
3	Total no. of Test cases written for all requirements	100
4	Total no. of Test cases Executed	65
5	No. of Test cases Passed	30
6	No. of Test cases Failed	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

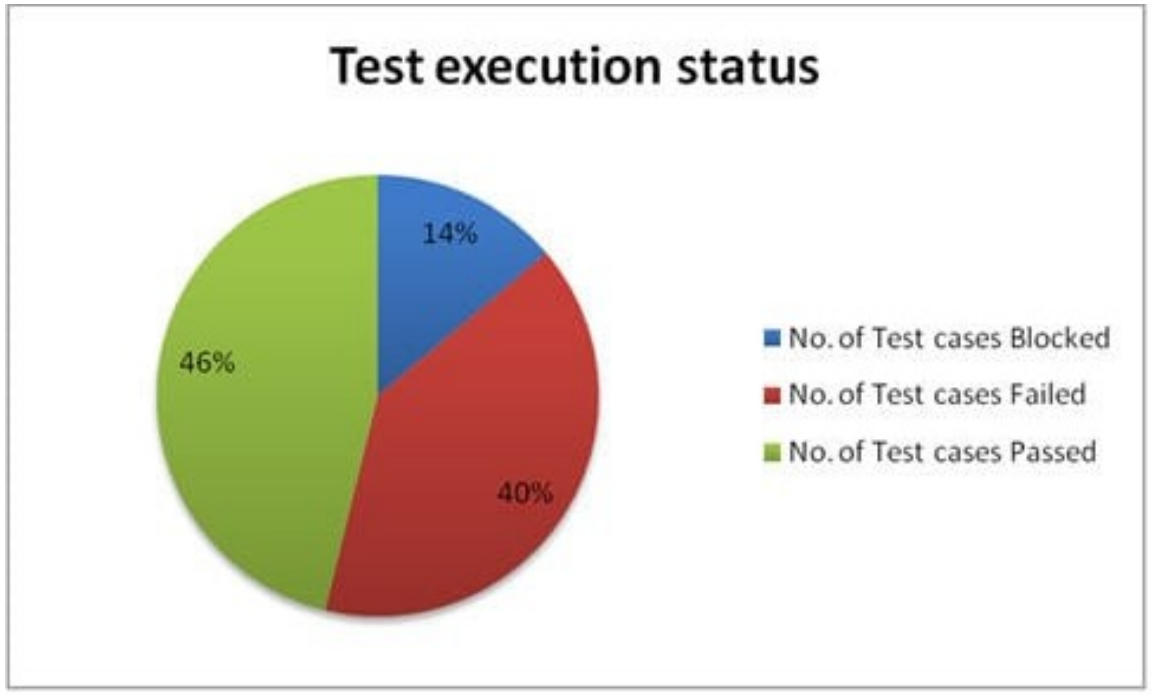
% High Defects = No. of High Defects identified / Total no. of Defects identified * 100

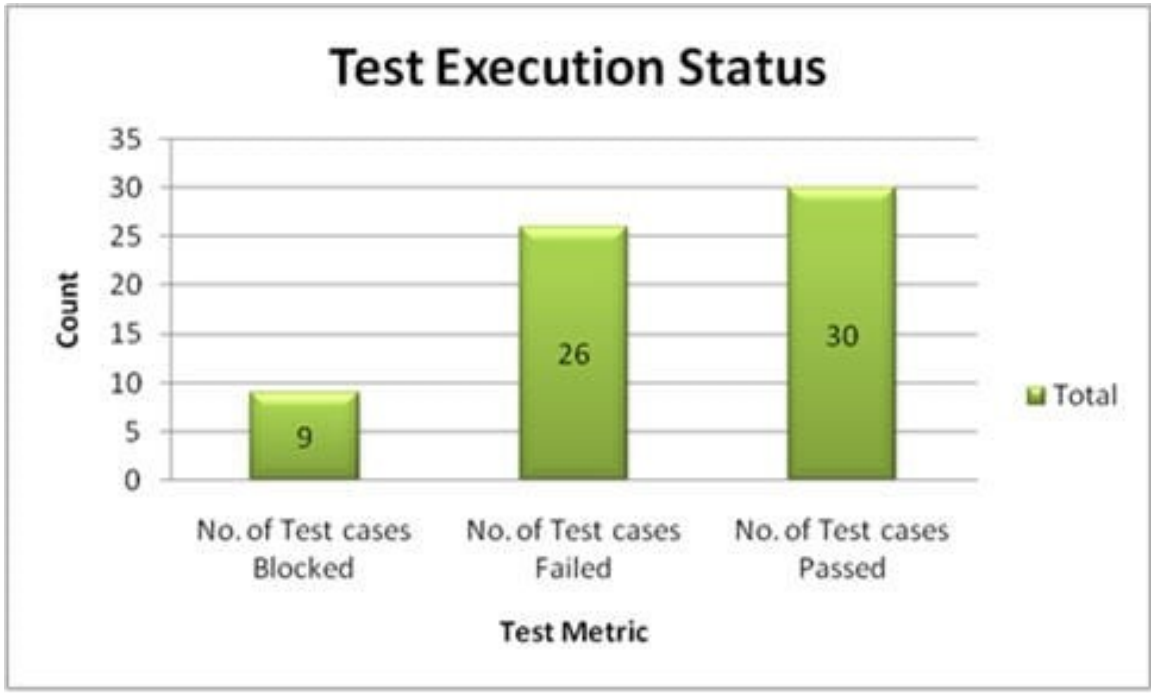
EXAMPLE:

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

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

Test Execution Examples





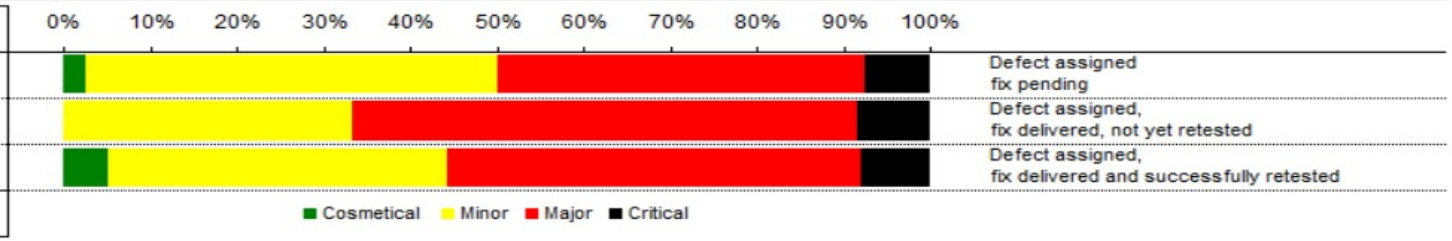
https://www.softwaretestinghelp.com/software-test-metrics-and-measurements/; NeilPatel.com

Software Development - Defect Dashboard

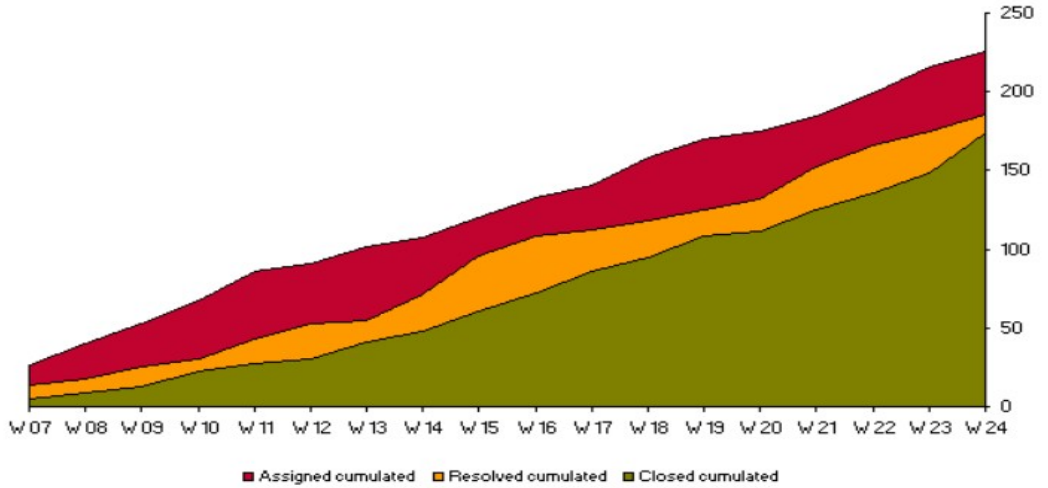
from Mo, 06/08/09 to So, 06/14/09 Week 24

Defects actual week by type and severity

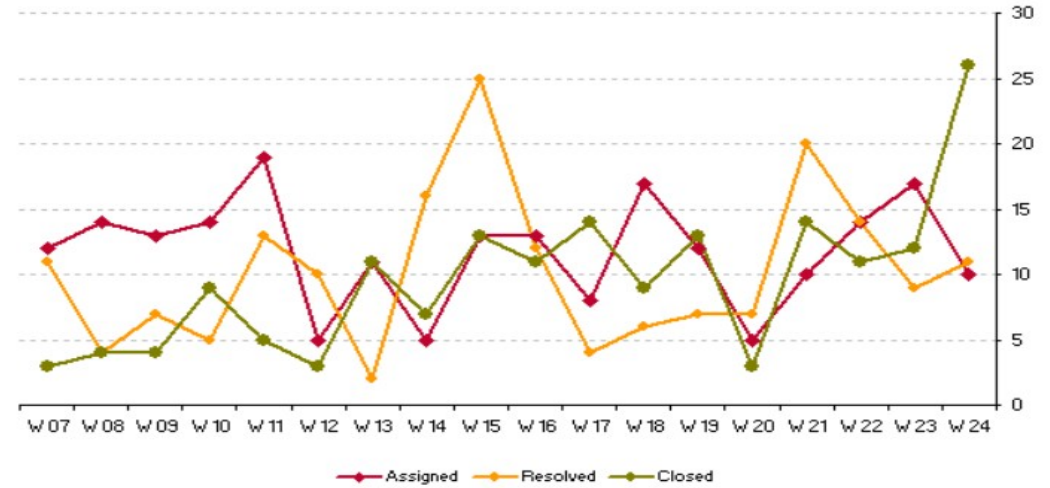
	Cosmetical	Minor	Major	Critical	Total
assigned	1	19	17	3	**40**
resolved	0	4	7	1	**12**
closed	9	68	83	14	**174**
Total	**10**	**91**	**107**	**18**	**226**



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

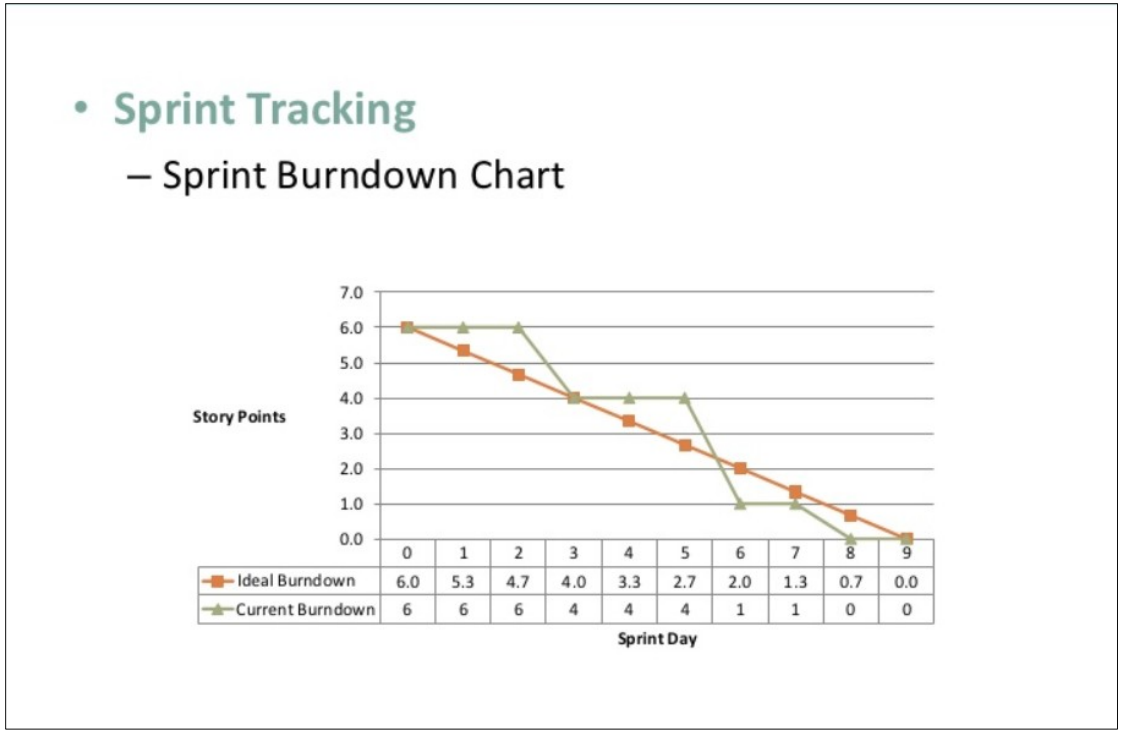
More QA Metrics…

- **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.

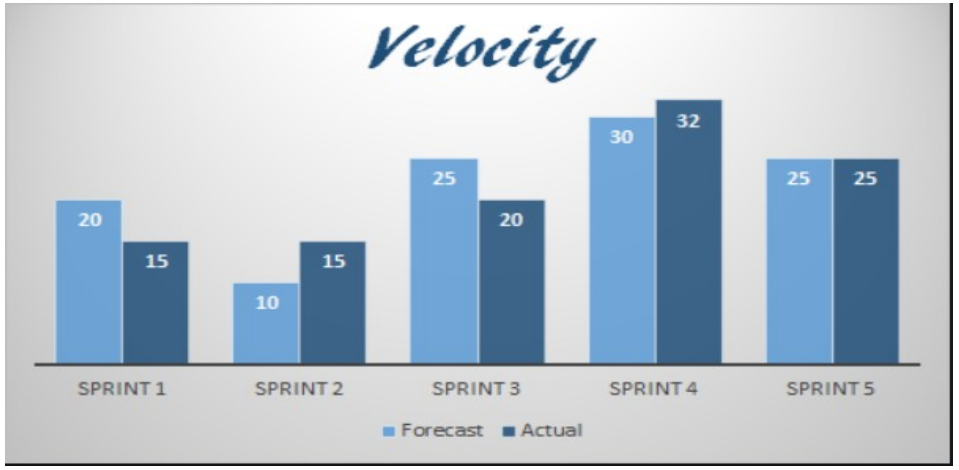
- **Sprint burndown chart:** A graph which shows how much work is remaining in the current iteration, and how much testing remains to be done.

- **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…

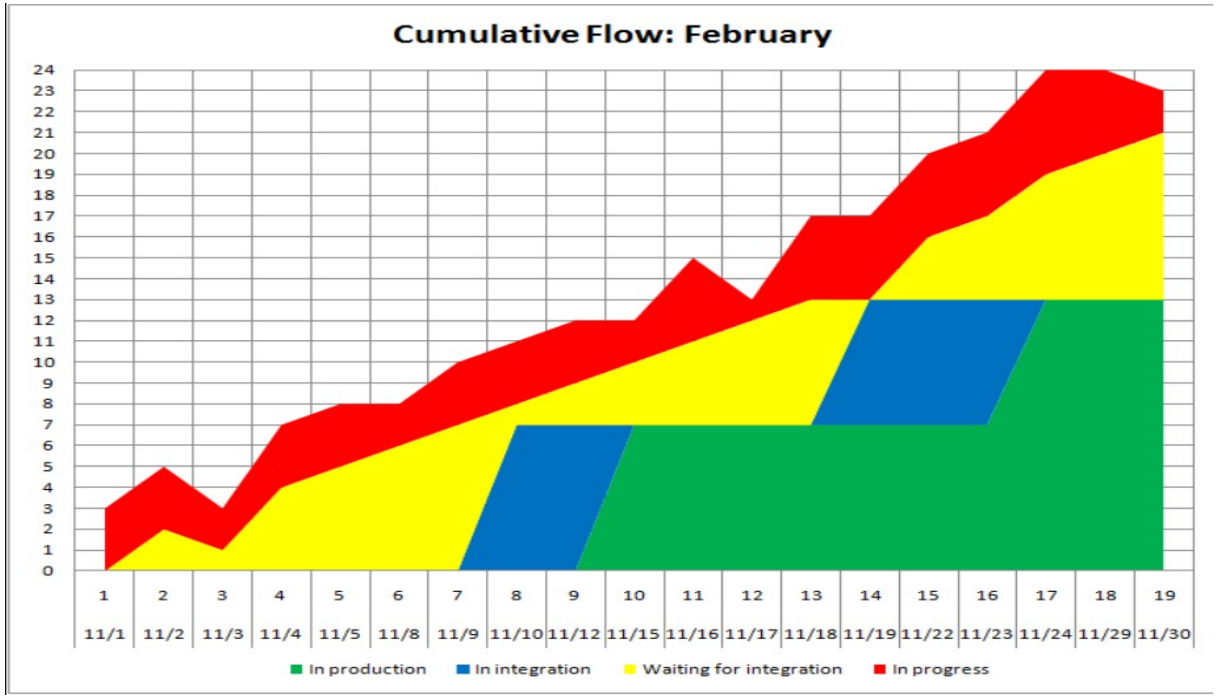
- **Number of working tested features / running tested features** – The more features or agile “stories” are consistently added to the software and fully tested, the healthier the project.
- **Velocity** – The **speed** at which the development team is completing new features.
  - It predicts how much 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