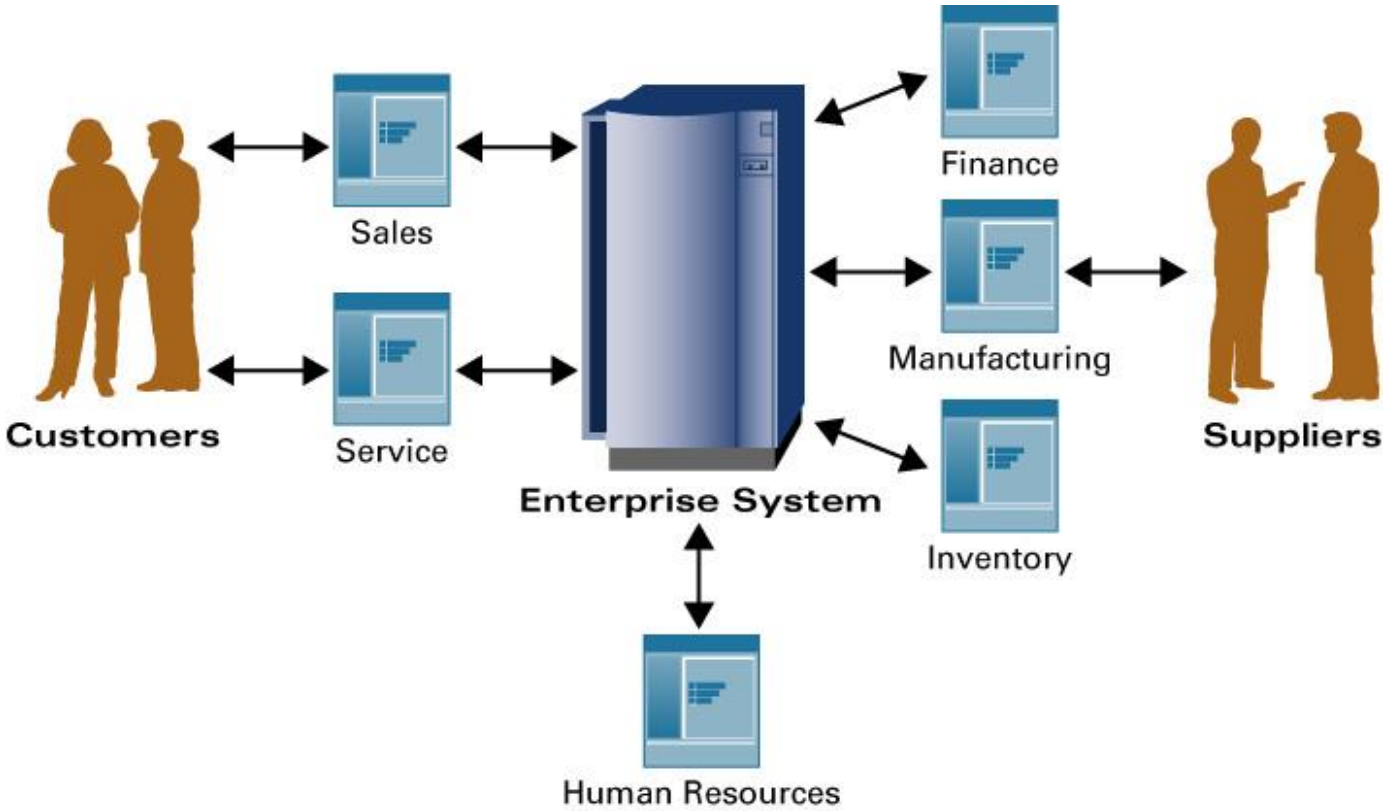
Software Testing Environment

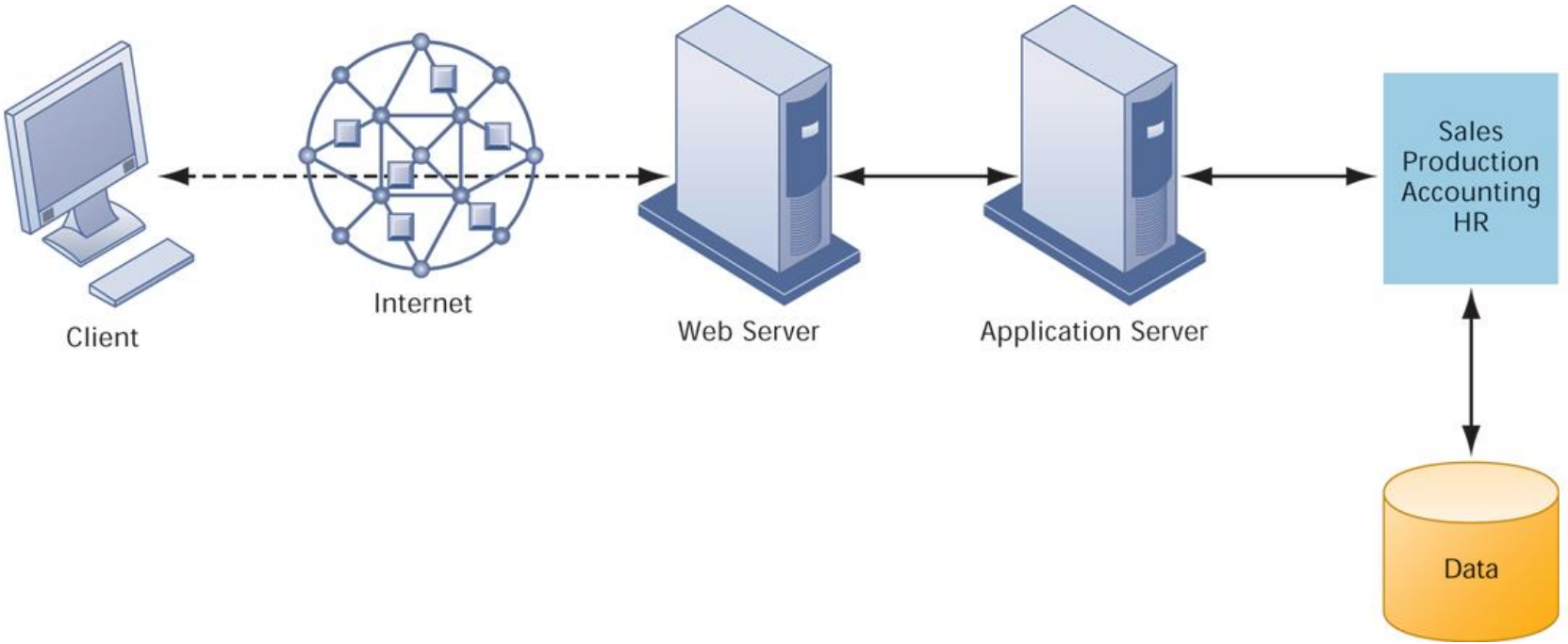
Nizam Mahmood
Infrastructure Architect

ORACLE Certified Professional
Microsoft CERTIFIED Systems Administrator
Microsoft CERTIFIED Database Administrator
Sun Microsystems CERTIFIED SOLARIS System Administrator
CISCO CERTIFIED CCNP
Certified for IBM WebSphere. software

Enterprise Systems



Enterprise Systems

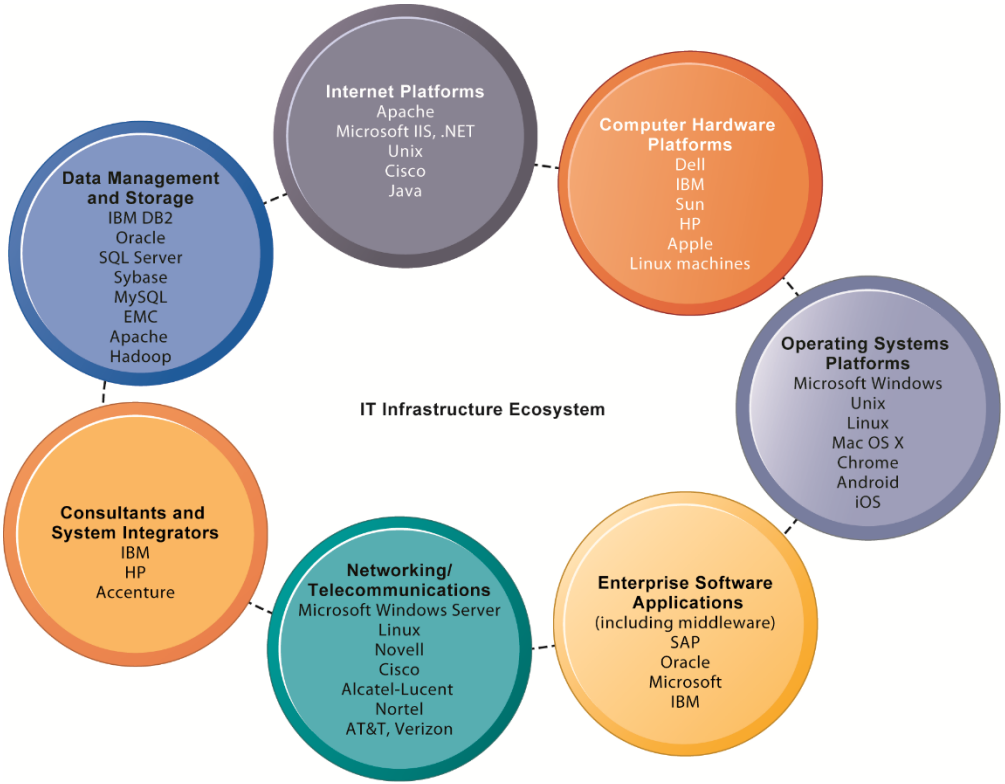


Infrastructure Components

- IT Infrastructure has seven main components
  1. Computer hardware platforms
  2. Operating system platforms
  3. Enterprise software applications
  4. Data management and storage
  5. Networking/telecommunications platforms
  6. Internet platforms
  7. Consulting system integration services

Infrastructure Components

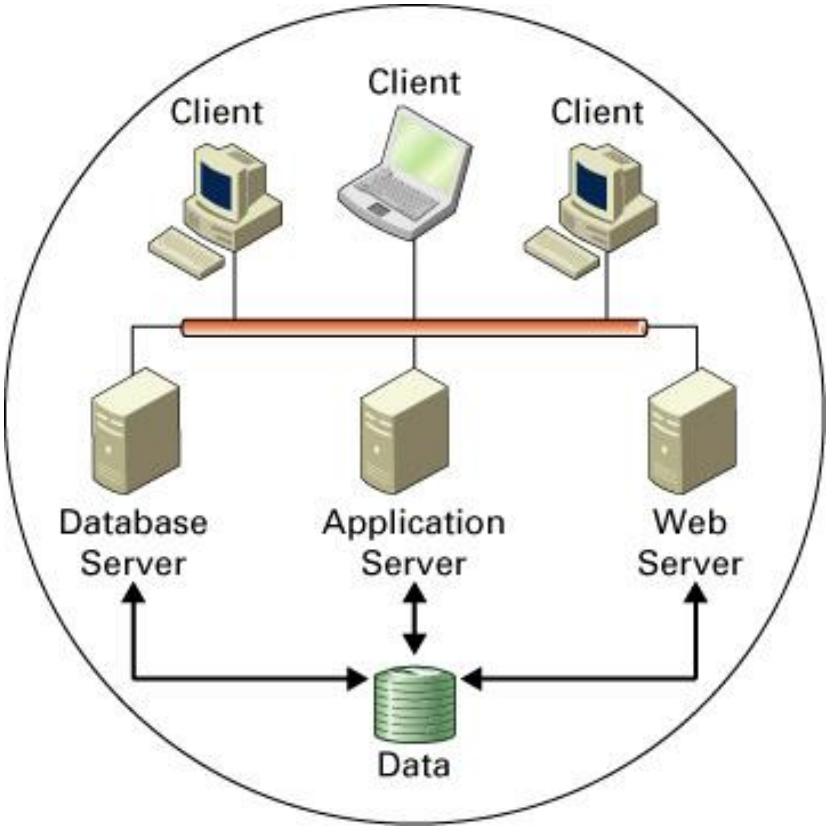
- IT Infrastructure has seven main components



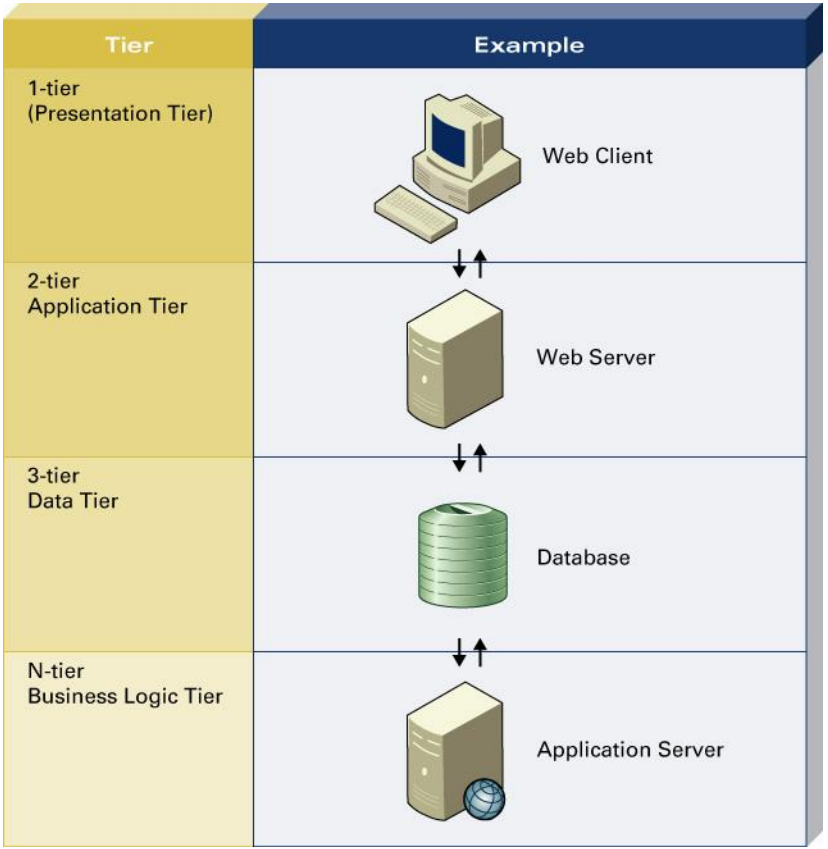
Client/Server Infrastructure

- A ***client/server infrastructure*** (or ***client/server network***) has one or more computers that are *servers* which provide services to other computers, called *clients*
- The client/server infrastructure is a form of distributed infrastructure
  - Application processing is split between the client and server
    - When surfing the web, your computer is the client using browser software and interacting with Web servers that have information you are seeking (shopping, news, education, etc.)
  - The server sends information to the client where it is processed – the network is heavily used which can become a bottleneck

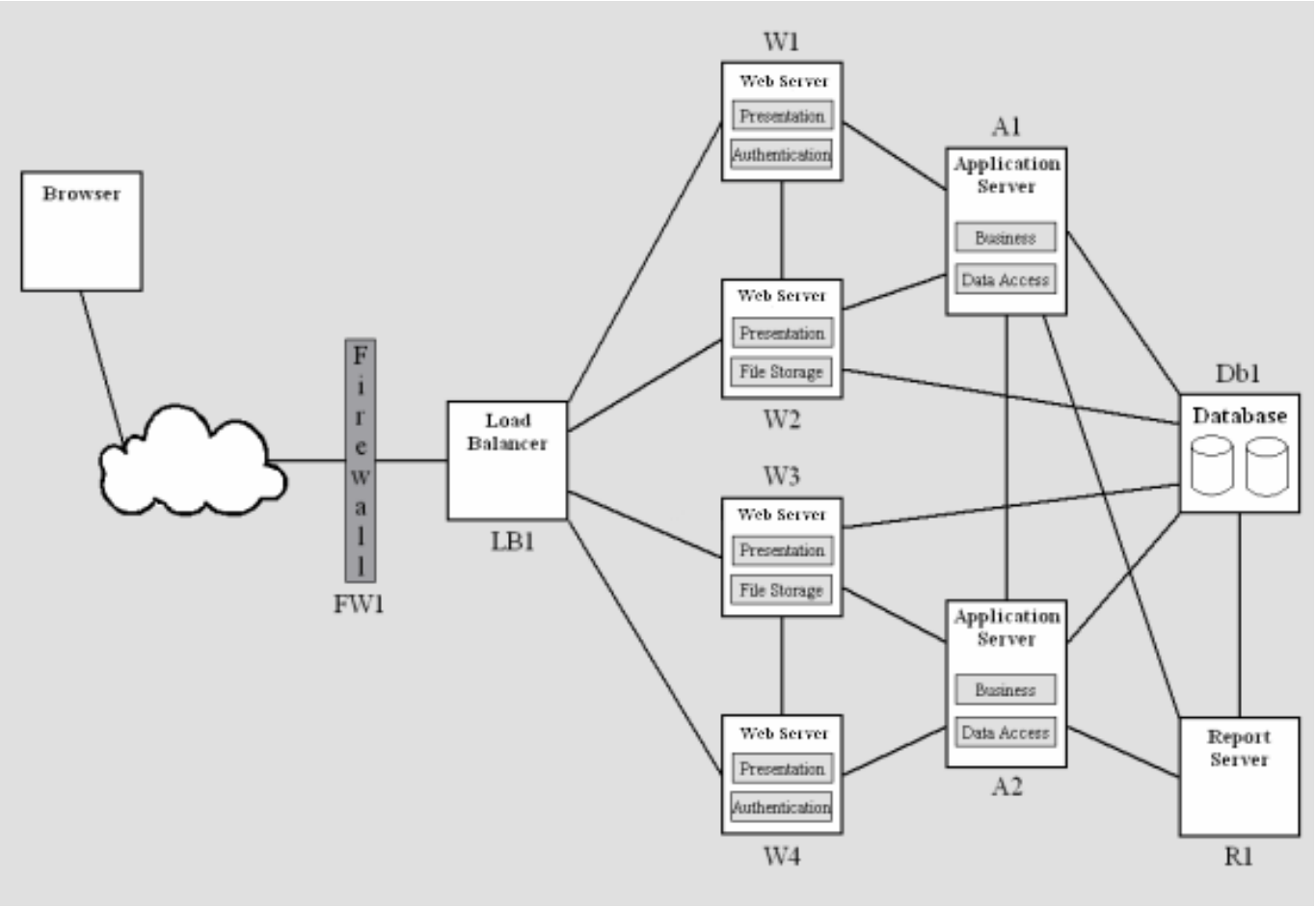
Client/Server Infrastructure



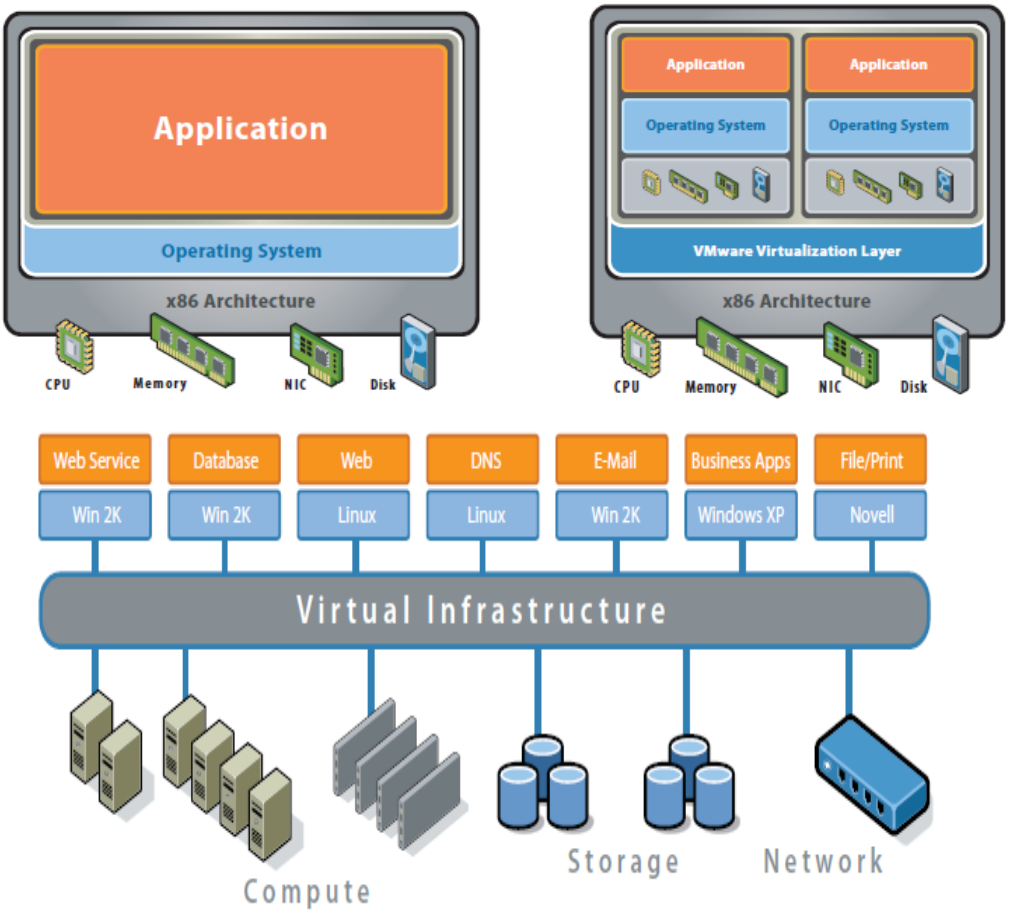
N-Tier Infrastructure



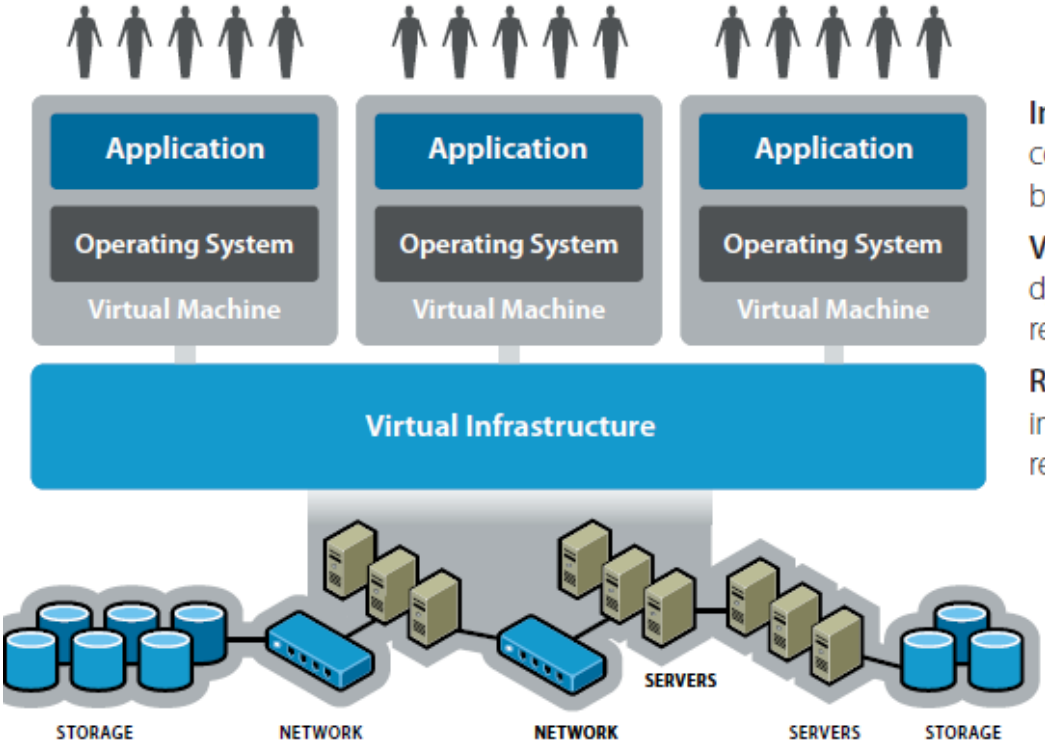
Enterprise Application



Virtualization Overview



Virtual Infrastructure

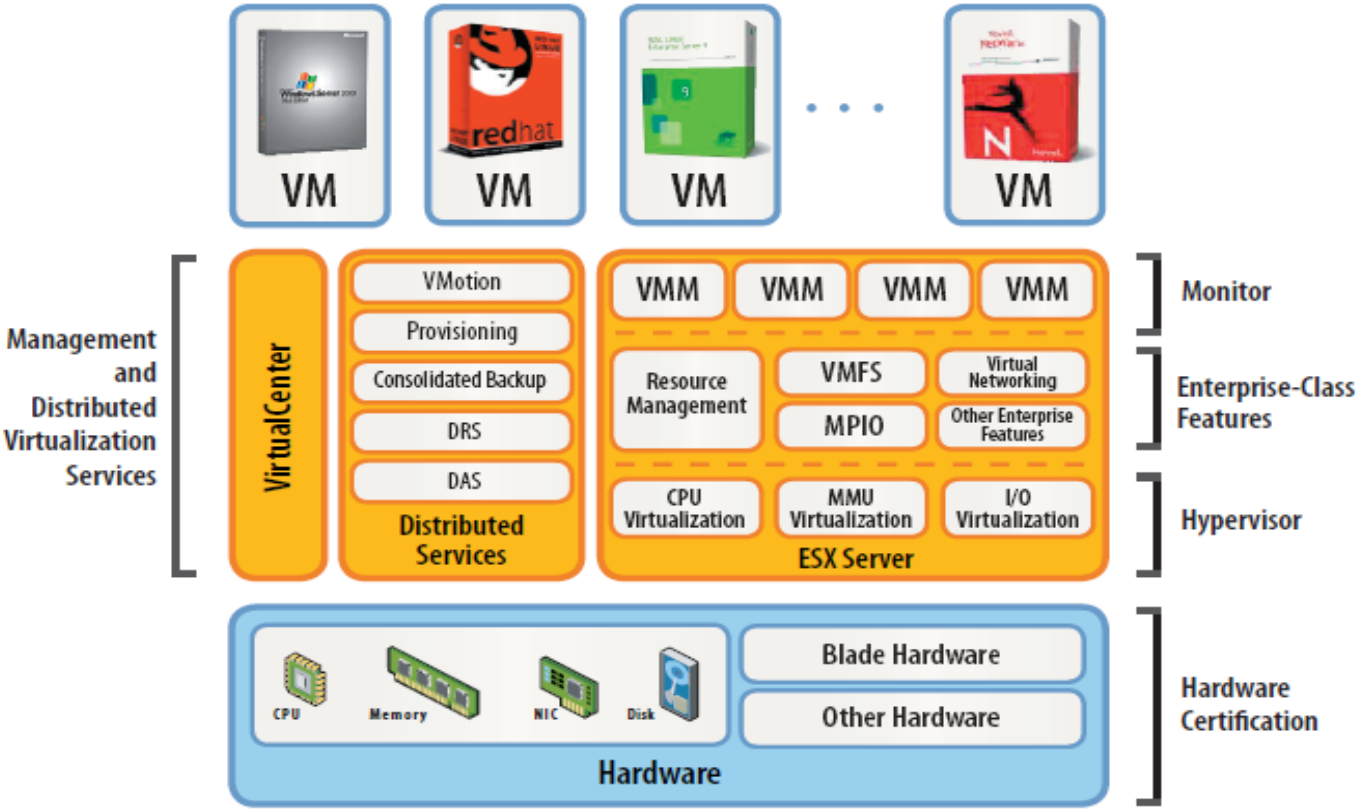


Infrastructure is what connects resources to your business.

Virtual Infrastructure is a dynamic mapping of your resources to your business.

Result: decreased costs and increased efficiencies and responsiveness

Virtual Infrastructure



Cloud Computing

Cloud computing, often referred to as simply “the cloud,” is the delivery of on-demand computing resources—everything from applications to data centers—over the Internet on a pay-for-use basis

- Infrastructure as a service
- Platform as a service
- Software as a service

- **Cloud can be public or private**
- **Allows companies to minimize IT investments**
- **Drawbacks: Concerns of security, reliability**
- **Hybrid cloud computing model**

Software as a service (SaaS)

Cloud-based applications—or software as a service—run on distant computers “in the cloud” that are owned and operated by others and that connect to users’ computers via the Internet and, usually, a web browser.

The benefits of SaaS :

- You can sign up and rapidly start using innovative business apps
- Apps and data are accessible from any connected computer
- No data is lost if your computer breaks, as data is in the cloud
- The service is able to dynamically scale to usage needs

Platform as a service (PaaS)

Platform as a service provides a cloud-based environment with everything required to support the complete lifecycle of building and delivering web-based (cloud) applications—without the cost and complexity of buying and managing the underlying hardware, software, provisioning, and hosting.
The benefits of PaaS

- Develop applications and get to market faster
- Deploy new web applications to the cloud in minutes
- Reduce complexity with middleware as a service

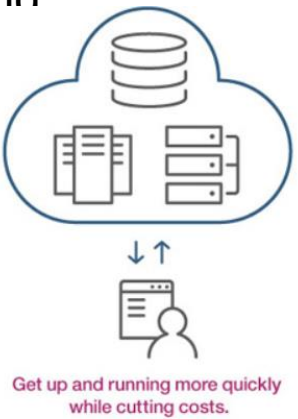
Deploy and migrate applications to both public and private clouds.

Infrastructure as a service (IaaS)

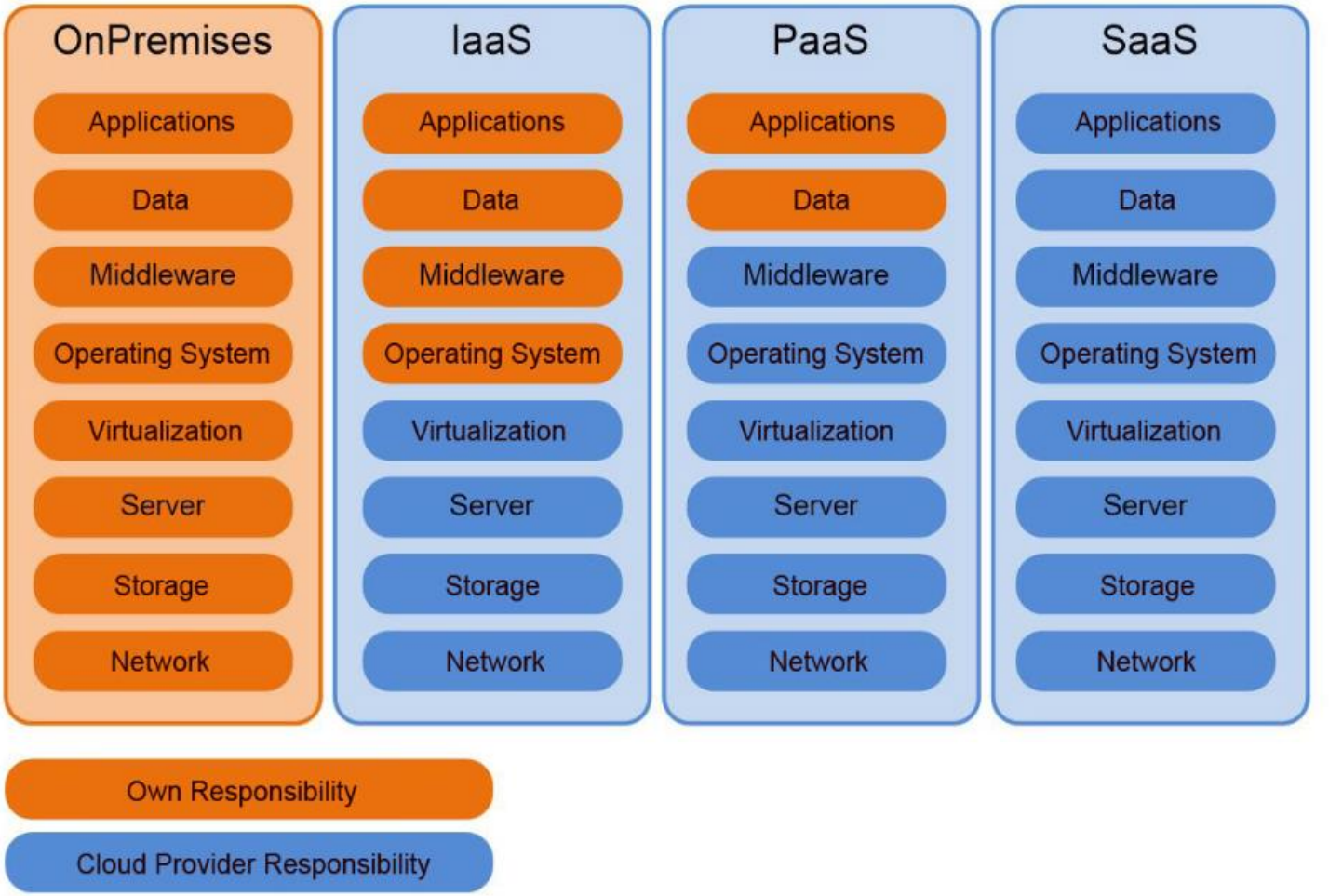
Infrastructure as a service provides companies with computing resources including servers, networking, storage, and data center space on a pay-per-use basis.

The benefits of IaaS

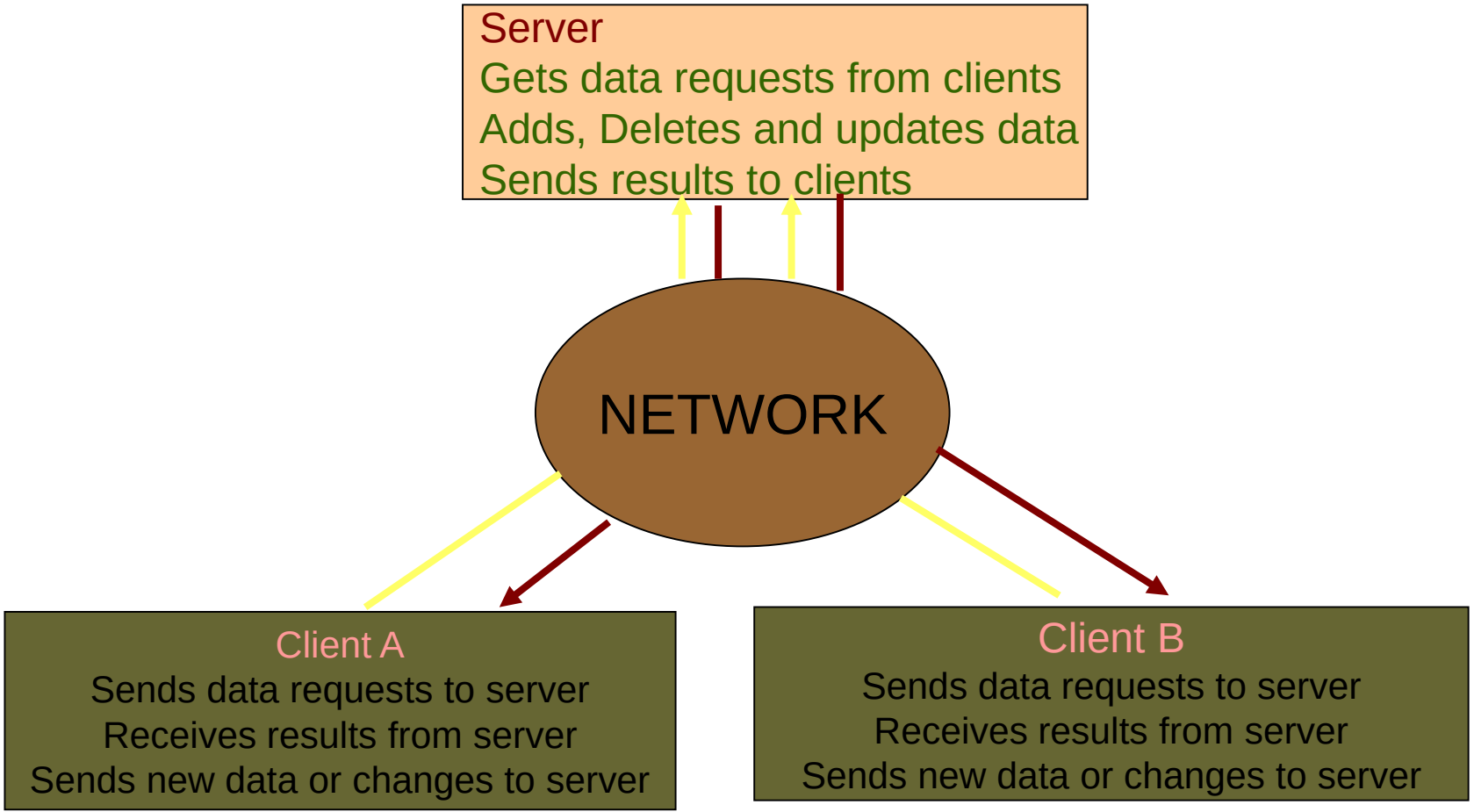
- No need to invest in your own hardware
- Infrastructure scales on demand to support dynamic workloads
- Flexible, innovative services available on demand



Cloud Computing



Client/server DBMS

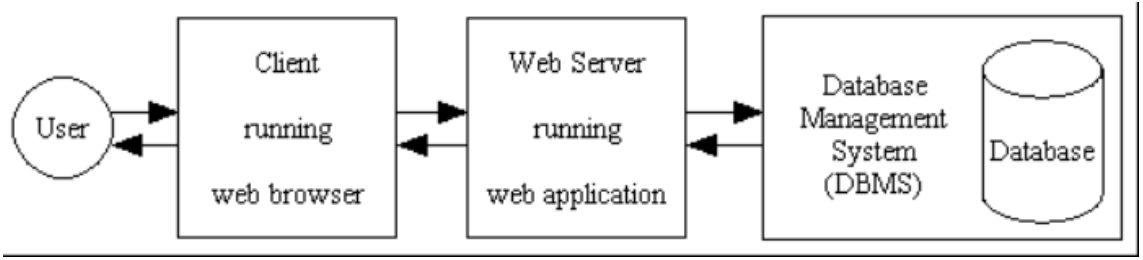


Application Layers

Presentation logic : The user interface (UI) which displays data to the user and accepts input from the user. In a web application this is the part which receives the HTTP request and returns the HTML response.

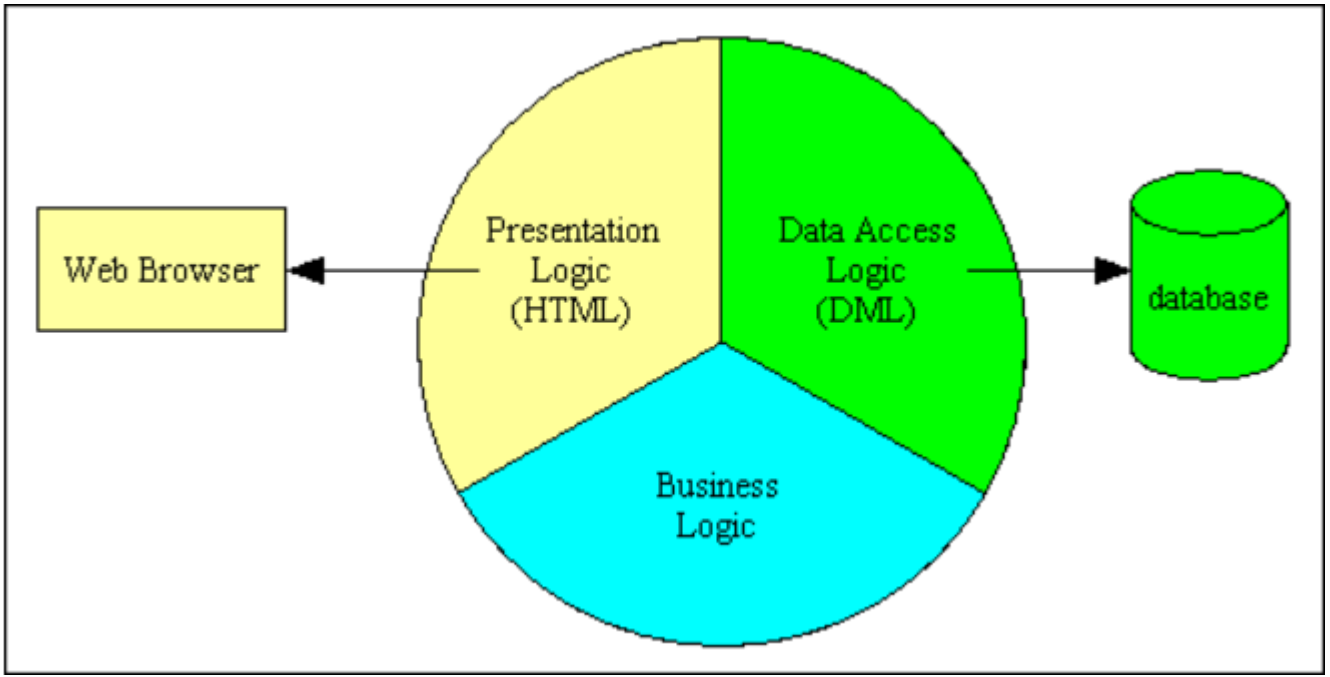
Business logic : Handles data validation, business rules and task-specific behavior.

Data Access logic : communicates with the database by constructing SQL queries and executing them via the relevant API.

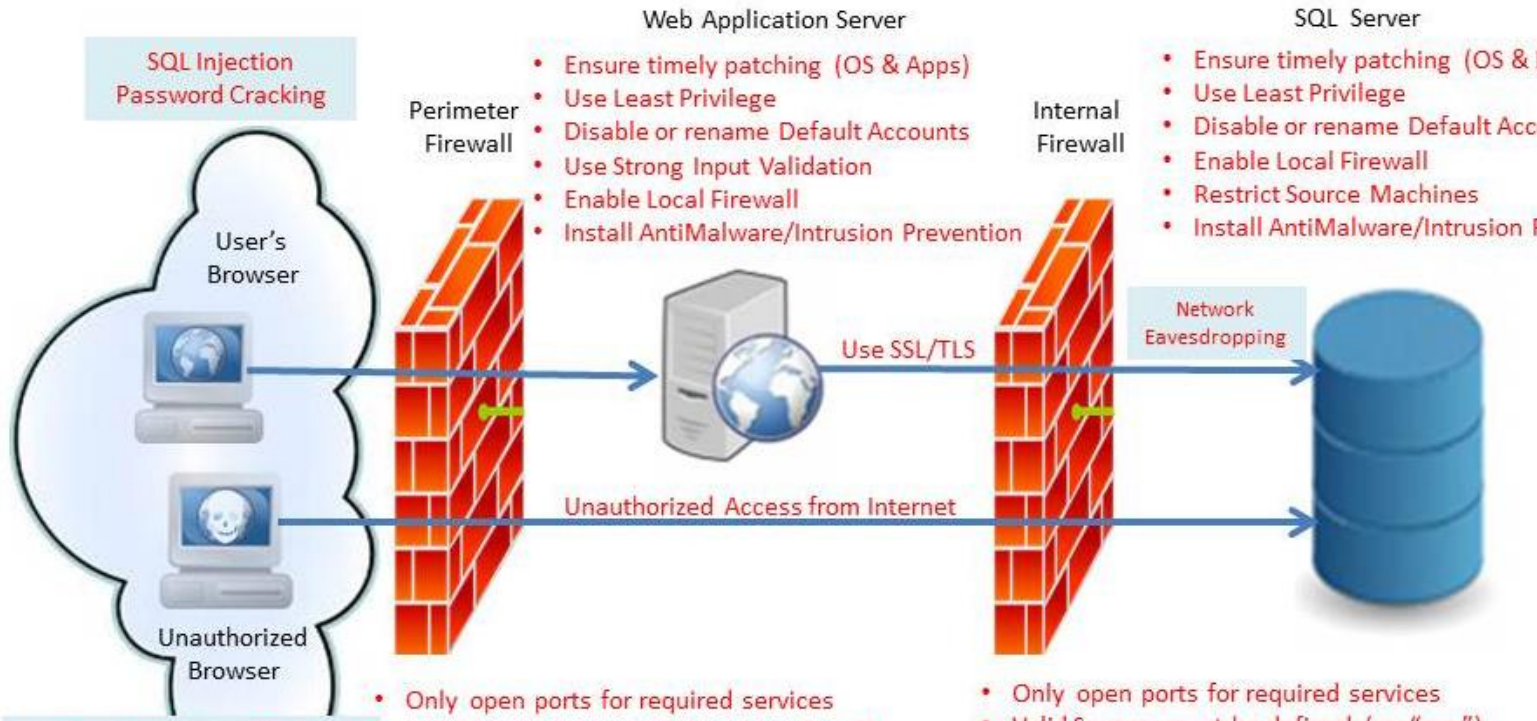


Application Layers

1 Tier Architecture



Backend Security



Overview of Database

A database is any collection of related data
A database is an organized collection of data

- A database can store the correctly
- A database can retrieve the data efficiently

DBMS

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. DBMS should be considered as a multi-layered system:

Database management systems provide several functions in addition to simple file management:

- **allow concurrency**
- **control security**
- **maintain data integrity**
- **provide for backup and recovery**
- **control redundancy**
- **allow data independence**
- **provide non-procedural query language**
- **perform automatic query optimization**

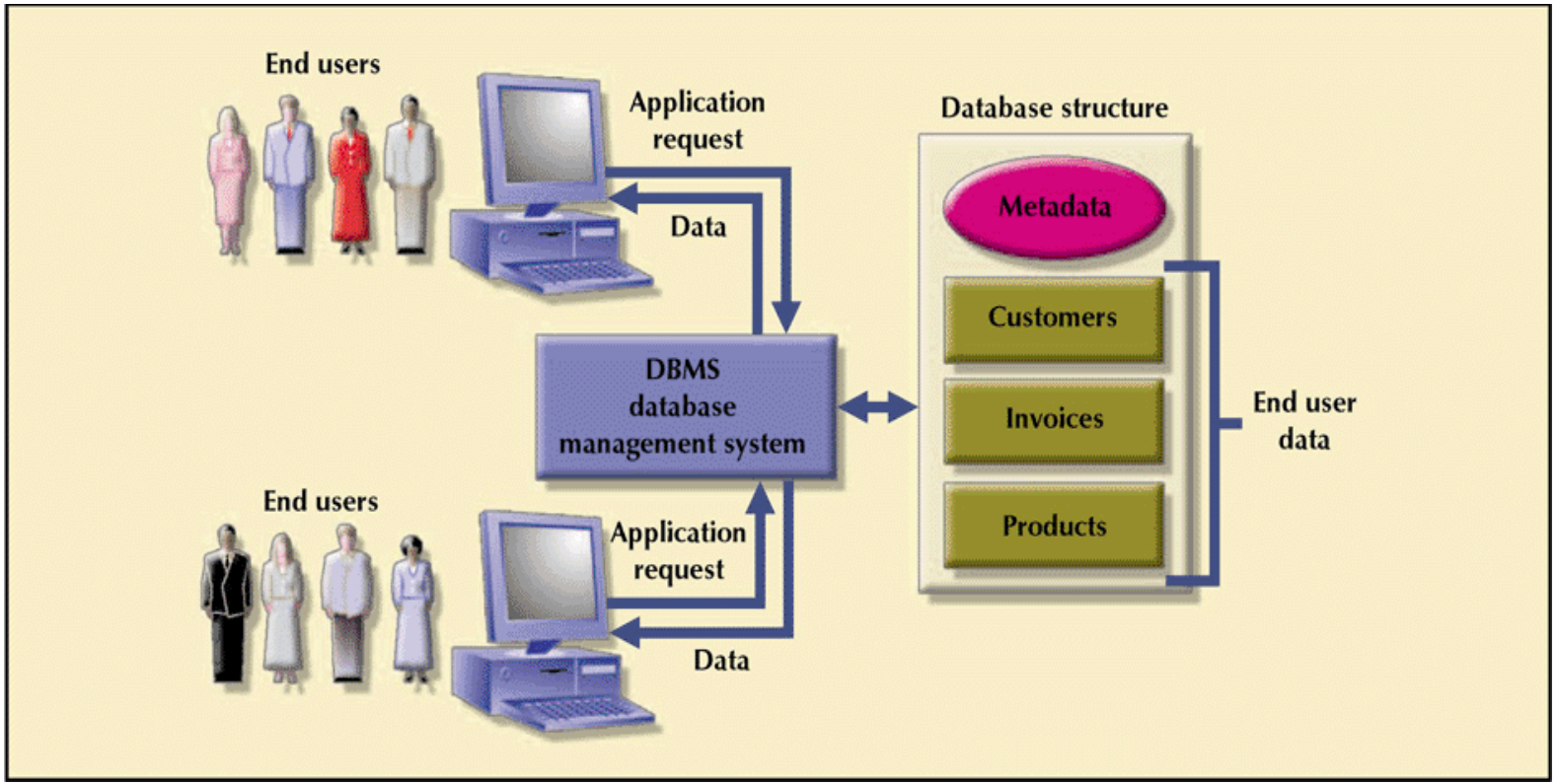
Overview of Database

In the early days, database applications were built on top of file systems

Drawbacks of using file systems to store data:

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation — multiple files and formats
- Integrity problems
  - Integrity constraints (e.g. account balance > 0) become part of program code
  - Hard to add new constraints or change existing ones

interaction between end users and database



Relational Database Server Goals

- Reliability
- Availability
- Scalability
- Performance
- Data Integrity and Protection
- Transaction Isolation
- Reporting
- Data Analysis

Silos in the IT Lifecycle

Silos in the IT Lifecycle

Infrastructure Architect

Solution Architect

Developer

Tester

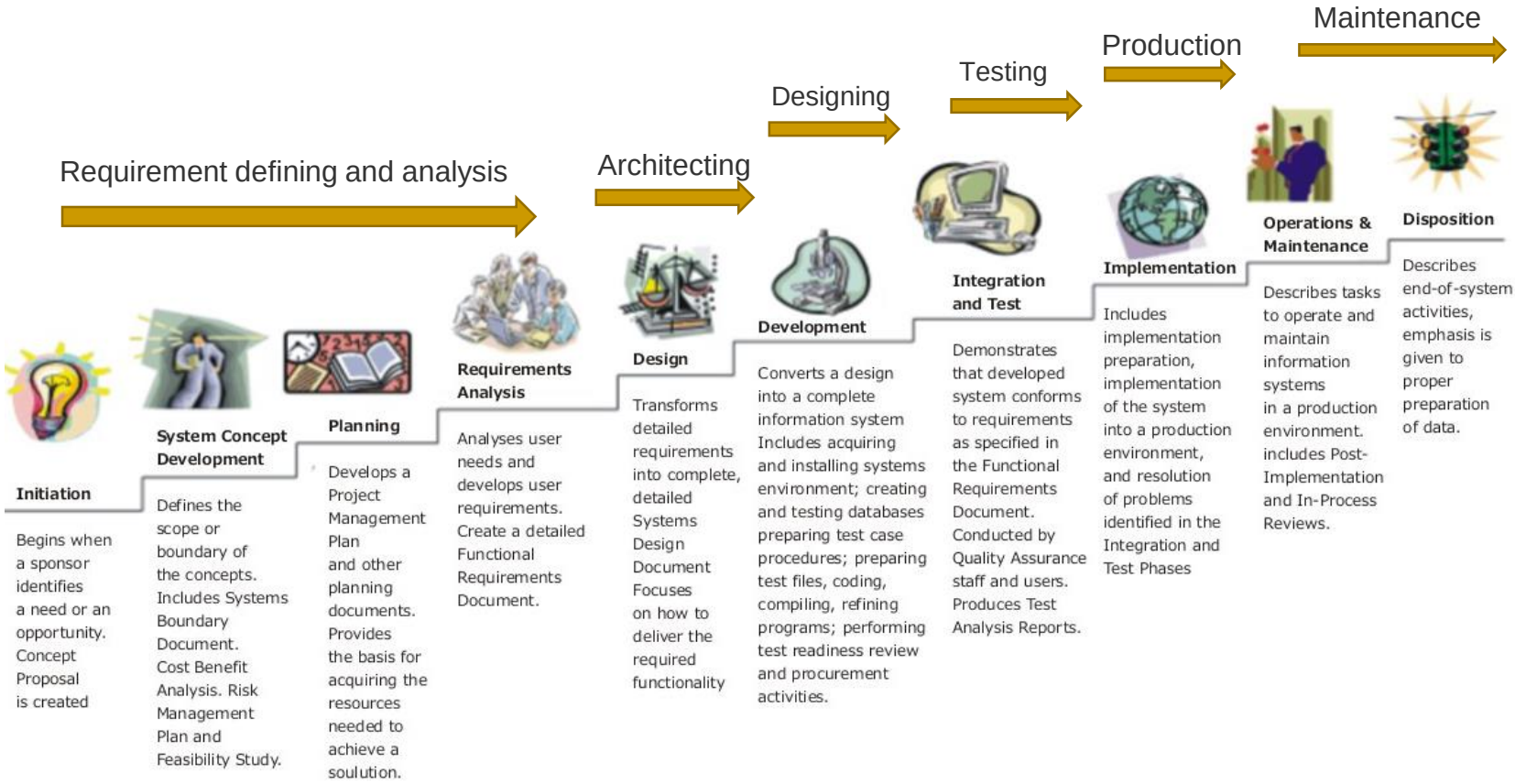
Project Manager

End User

SDLC


Business Analysis
Requirement Gathering
Architecting & Design
Development
Testing & Evaluation
Deployment
Operational & Maintenance

SDLC


Requirement defining and analysis
Architecting
Designing
Testing
Production
Maintenance
Initiation
Begins when a sponsor identifies a need or an opportunity. Concept Proposal is created
System Concept Development
Defines the scope or boundary of the concepts. Includes Systems Boundary Document. Cost Benefit Analysis. Risk Management Plan and Feasibility Study.
Planning
Develops a Project Management Plan and other planning documents. Provides the basis for acquiring the resources needed to achieve a soulution.
Requirements Analysis
Analyses user needs and develops user requirements. Create a detailed Functional Requirements Document.
Design
Transforms detailed requirements into complete, detailed Systems Design Document Focuses on how to deliver the required functionality
Development
Converts a design into a complete information system Includes acquiring and installing systems environment; creating and testing databases preparing test case procedures; preparing test files, coding, compiling, refining programs; performing test readiness review and procurement activities.
Integration and Test
Demonstrates that developed system conforms to requirements as specified in the Functional Requirements Document. Conducted by Quality Assurance staff and users. Produces Test Analysis Reports.
Implementation
Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phases
Operations & Maintenance
Describes tasks to operate and maintain information systems in a production environment. includes Post-Implementation and In-Process Reviews.
Disposition
Describes end-of-system activities, emphasis is given to proper preparation of data.

SDLC

Waterfall Model

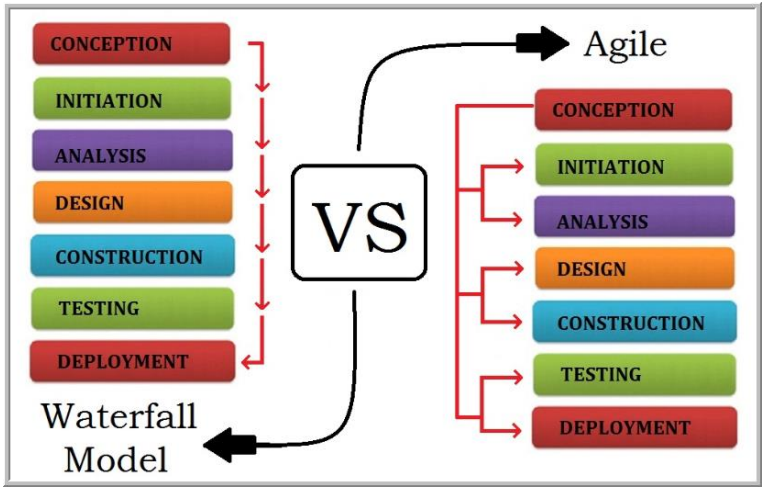
- Linear model of software design.
- Waterfall employs a sequential design process. Development flows sequentially from start point to end point, with several different stages: Conception, Initiation, Analysis, Design, Construction, Testing, Implementation, and Maintenance.

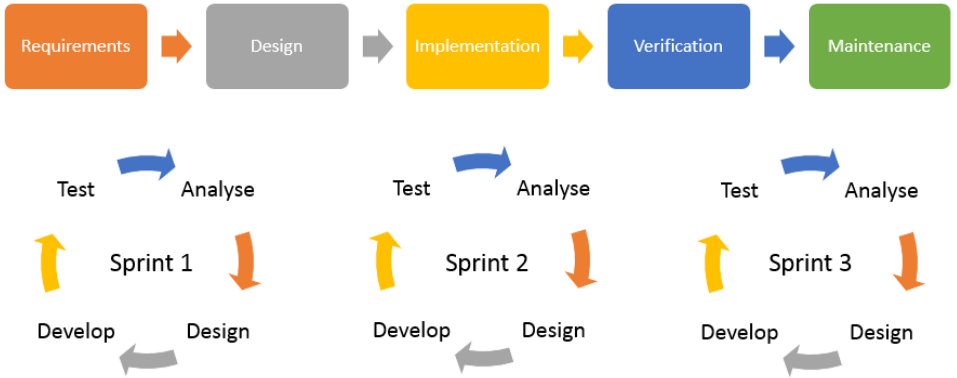
Agile Model

- Agile proposes an incremental and iterative approach to software design.
- There is no pre-determined course of action or plan with the Agile method
- Lightweight
- People-based rather than Plan-based

SDLC

Waterfall Model vs Agile Model:





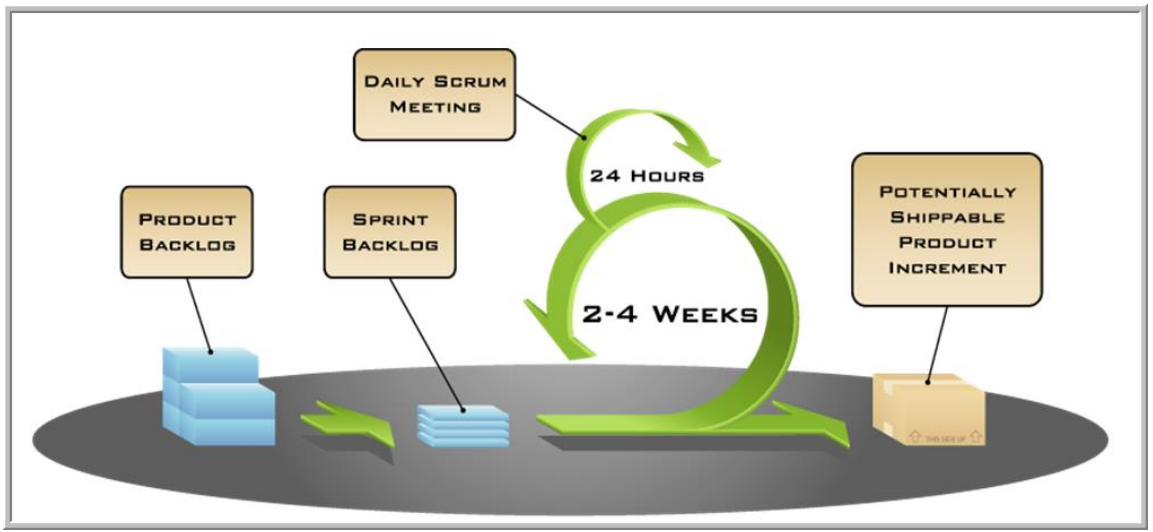
SDLC

SCRUM:

- Scrum is an agile process that allows to focus on delivering the highest business value in the shortest time.
- It allows rapidly and repeatedly inspect actual working software (every two weeks to one month).
- The business sets the priorities. Development team to determine the best way to deliver the highest priority features.
- Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance for another iteration.

SDLC

SCRUM:



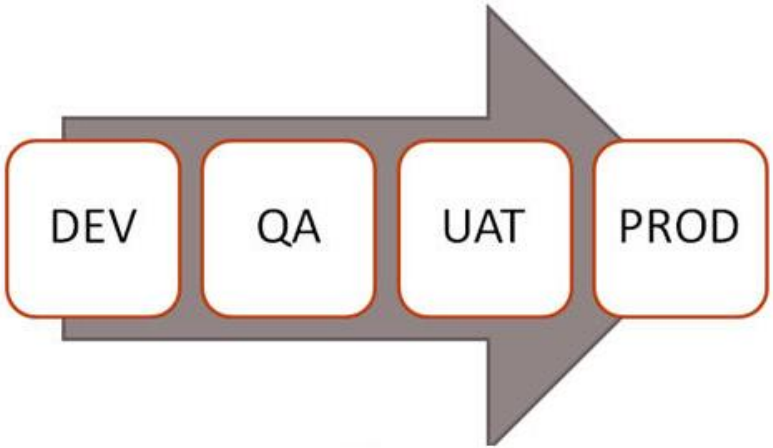
SCRUM Characteristics :

- Self-organizing teams
- Product progresses in a series of month-long **“sprints”**
- Requirements are captured as items in a list of **“product backlog”**
- No specific engineering practices prescribed

Software Development Process

Typical development lifecycle:

1) Create development environments.
2) Create testing environments, including UAT and integration.
3) Migrate changes from development environment to integration environment.
4) Regrassion Testing.
5) Migrate changes from integration environment to UAT environment.
6) Perform user-acceptance tests.
7) Release changes to Production
8) End user Test



Version Control

Version control system is software tool that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

- Multiple developers working on same code
- Access to older versions of files
- Change log
- Comparison between different versions
- Multiple versions at same time

Version Control

What is Repository

Repository stores a complete copy of all the files and directories which are under version control

- Normally, whenever access any of the files in the repository directly. Instead, we use commands to get your own copy of the files into a "working directory", and then work on that copy.
- When we are finished a set of changes, we check (or "commit") them back into the repository
- Repository and Working directory are totally separate

What is Checkout

- Copying a module(s) or file(s) from the server to the local directory(working directory)is called a *checkout*
- It will create a local copy of files
- These copies are latest versions of the files available in server

What is Check-in

- After checkout and update of a file is done that file is ‘check in’ back into to the repository
- This will create a new version of a file in the repository
- Comments can be added at the check in time to specify reasons of modification
- Some tools require ‘commit’ command to be executed after check in

Software Build Process

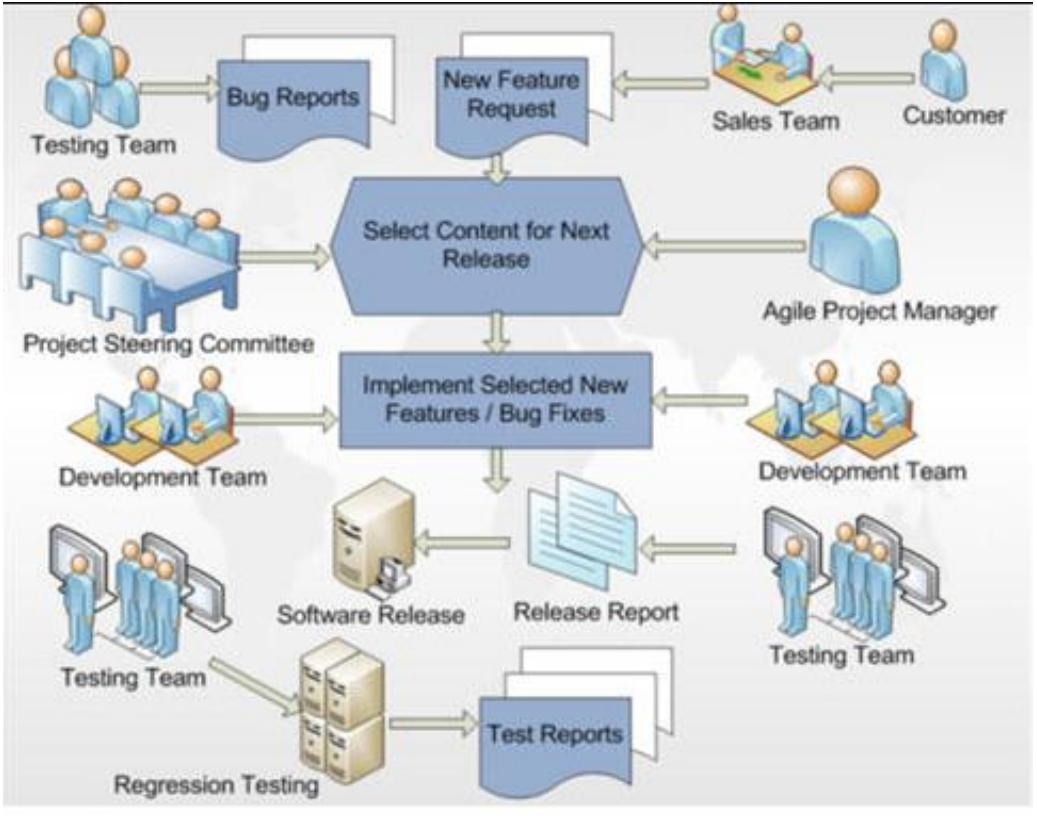
What is a Build?

Build is a process by which source code is converted into a stand-alone form that can be run on a computer.

One of the most important steps of a software build is the compilation process, where source code files are converted into executable code. The process of building software is usually managed by a build tool. Builds are created when a certain point in development has been reached or the code has been deemed ready for implementation, either for testing or release.

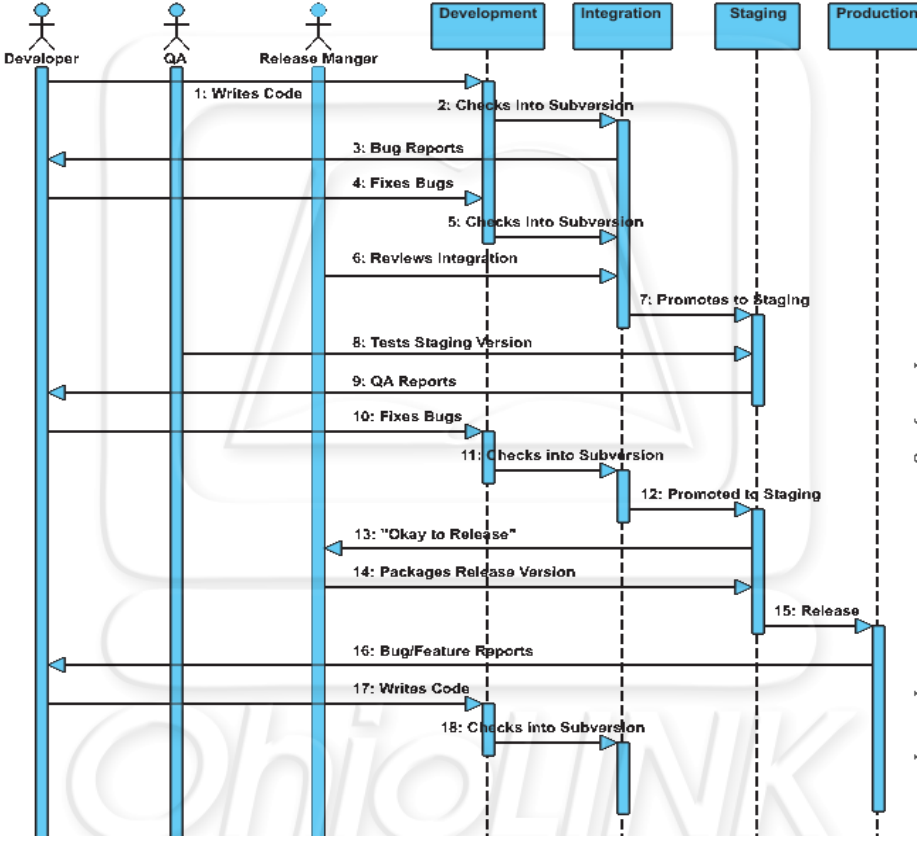
A build is also known as a software build or code build.

Code Release Process

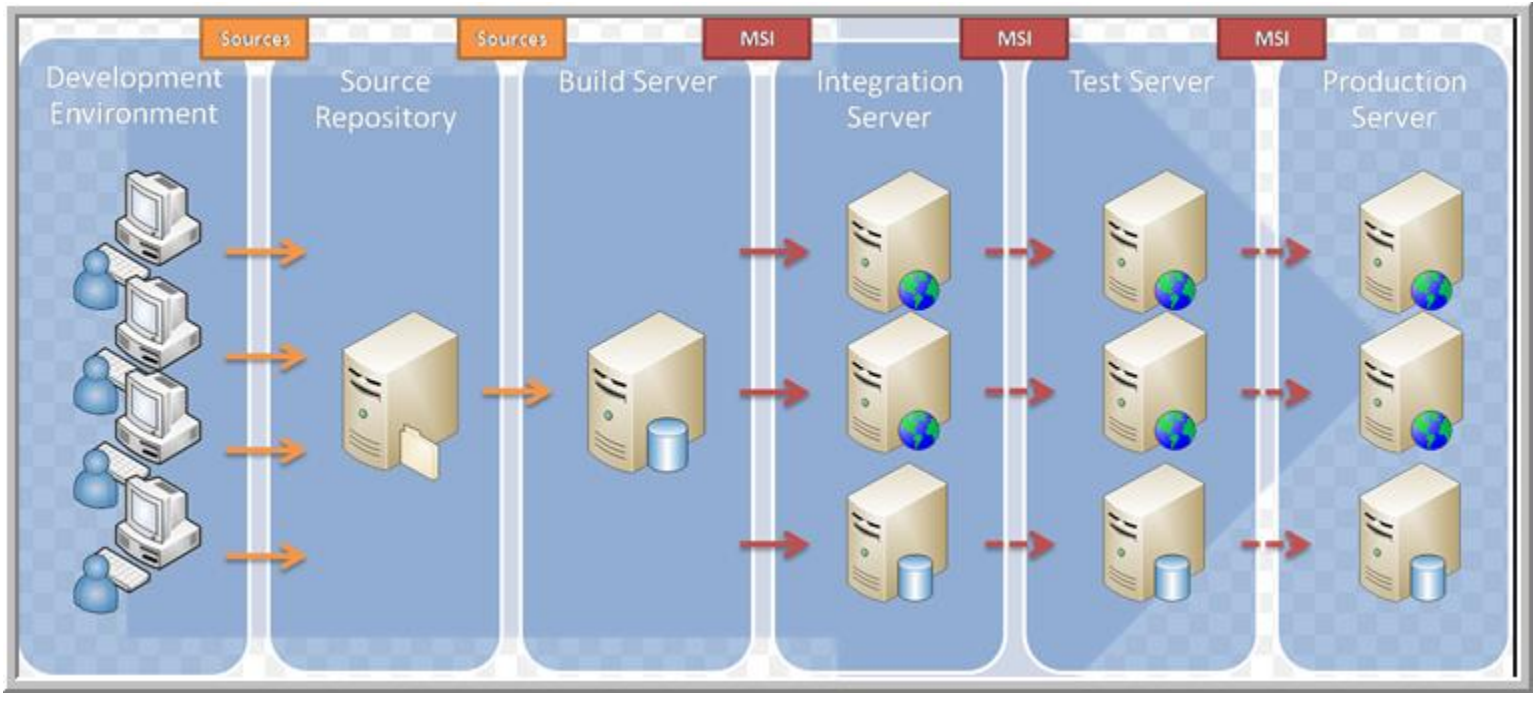


Test Release Content

Code Release Process



Software Build Process



What is a Bug ?

A bug is the error/fault of the software which is detected in the development environment before the product is deployed.

A defect is the error/fault of the software which is detected after it is deployed in the production environment.

When a defect is found in the production environment by the end/business users, they usually contact the Helpdesk to inform about the problem. Helpdesk collect as much information as possible about the defect and send it to a tester. The tester then tries to reproduce the defect and if it is a positive defect, it is eventually assigned to a developer to fix. Developer check out the codes from the source control, fixes the defect, performs unit testing and checked in the new codes to the source control. The build team then creates a new build and install it different environment. The QA tester test it and when they signed off, the build team make a new release and deploy it to the production environment.

Traceability matrix

A **traceability matrix** is a document, usually in the form of a table, used to assist in determining the completeness of a relationship by correlating any two baselined documents using a many-to-many relationship comparison.

Traceability matrix Benefits

- Track a requirement from conception through to delivery,
- Plan and manage testing and defect triages better,
- Reduce leakage, wastage of precious resource on non-priority, or simply non-requirements,
- Document adequately, and
- Work effectively in a world of integrations.

Black Box vs White Box Testing

Black Box Testing	White Box Testing
1. Black box testing techniques are also called functional testing techniques.	1. White box testing techniques are also called structural testing techniques.
2Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	2. White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
3. It is mainly applicable to higher levels of testing such as Acceptance Testing and System Testing	3. Mainly applicable to lower levels of testing such as Unit Testing and Integration Testing
4. Black box testing is generally done by Software Testers	4. White box testing is generally done by Software Developers
5. Programming knowledge is not required	5. Programming knowledge is required
6. Implementation knowledge is not required.	6. Implementation knowledge is required

Sanity vs smoke vs Regression Testing

Sanity Testing	Smoke Testing	Regression Testing
It is performed when a new functionality, change or bug fix is implemented.	It is performed in the initial phases when the release is unstable or at the final phase when the release is ready for deployment.	It is performed when a new functionality, change or bug fix is implemented.
It has a narrow scope.	It has only critical functionalities in the scope.	It has a broad scope.
The aim is to quickly verify if the new functionality, change or fix is working and has not broken down existing functionality.	The aim is to check if the critical functionalities are working as expected.	The aim is to check if the older functionalities are still working fine, after the change.
It does not catch all the bugs of the functional areas which are impacted by the code change.	It catches the bugs in critical functionalities only.	It catches all the bugs of the functional areas which are impacted by the code change.
It is non-scripted	It is scripted	It is scripted
It takes very less time to be performed.	It takes not more than 30 minutes.	It takes more time and testing effort.
It determines whether the application should be tested for regression.	It determines whether the application is stable or not.	It determines whether the old and new functionalities are working together correctly or not.

For additional questions or comments please post your topic on the forum at
www.peoplentech.com/forum
or send an email to: feedback@peoplentech.com